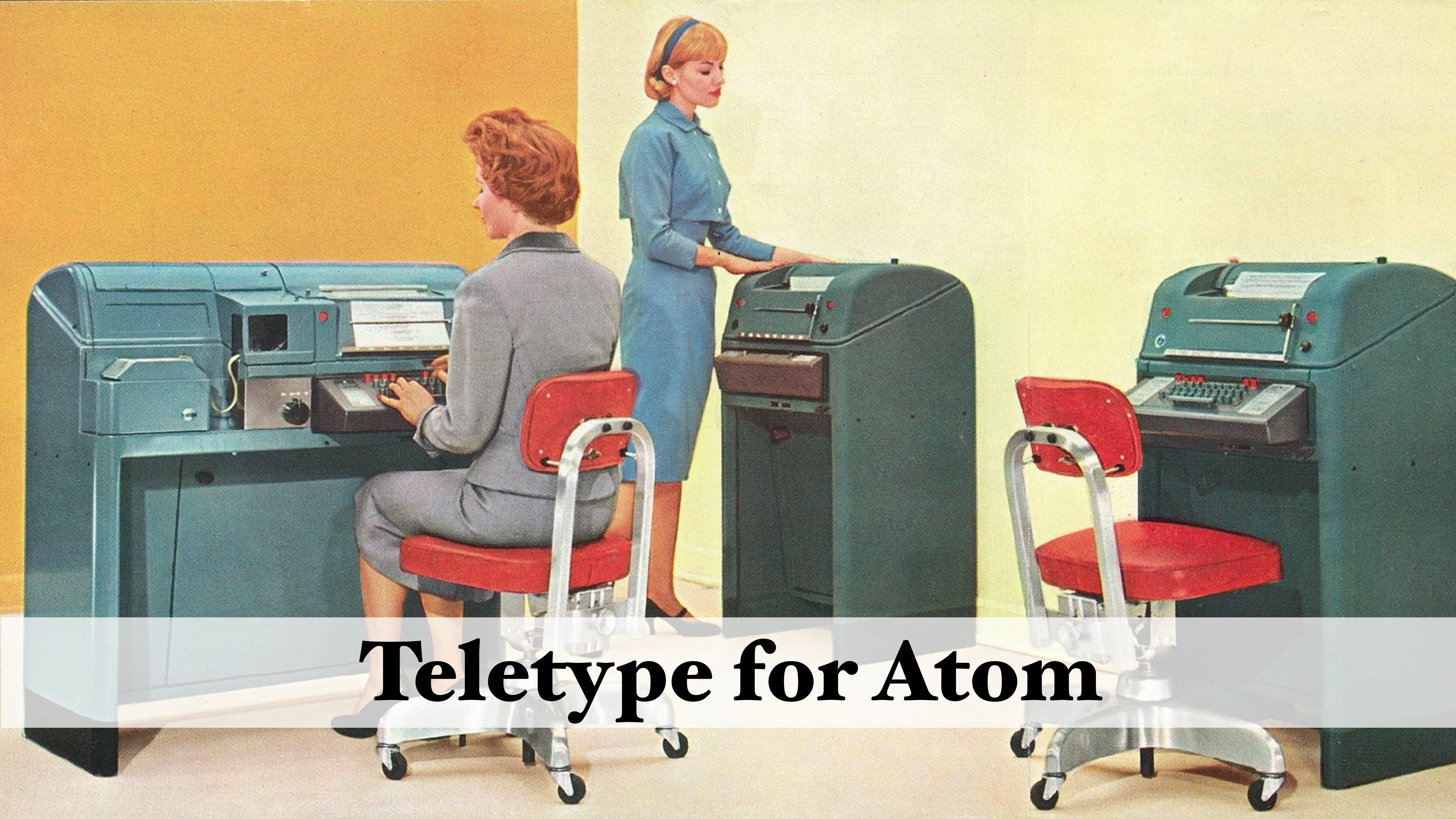


**Real-Time
Collaborative Editing
With CRDTs**



**Real-Time
Collaborative Editing
With CRDTs**

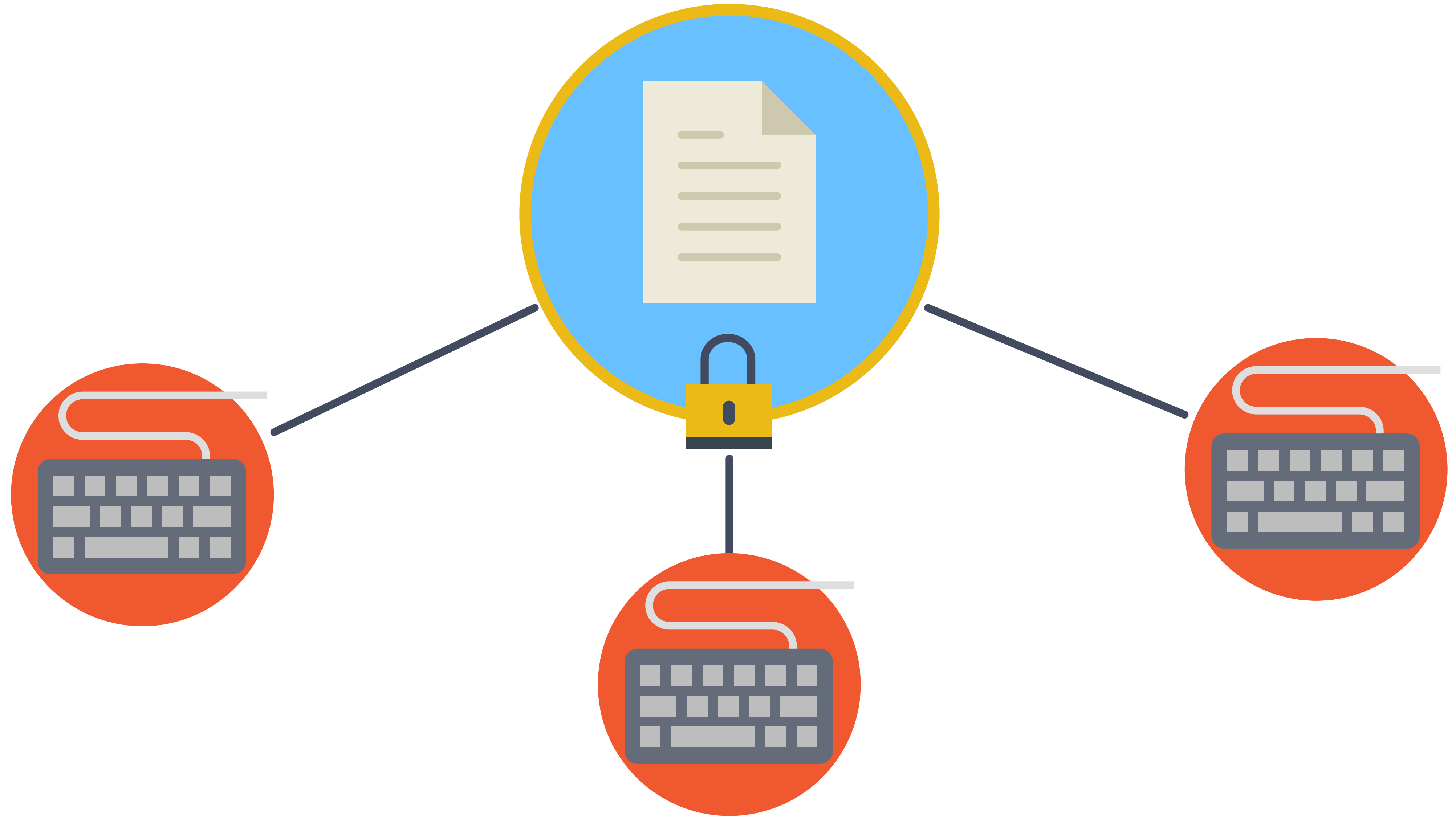



Teletype for Atom

**Make it as easy to code together
as it is to code alone.**

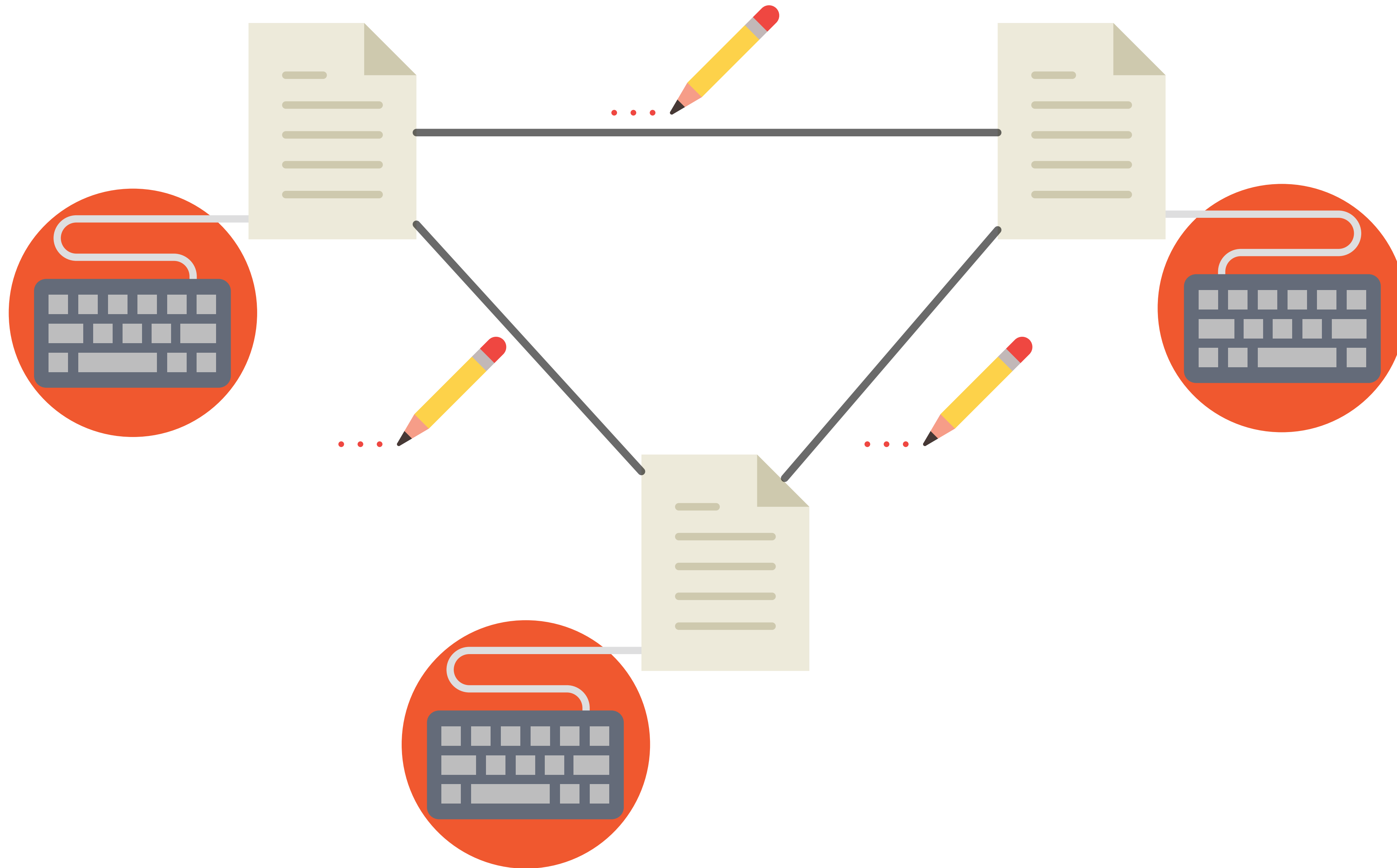
```
this.statusBar) {  
  arView()  
}
```

• LF LF UTF-8 Babel ^(**)  dev ↓ ↑  1 file





Boulder to Gambetta - 8795 km
~30ms at the speed of light



A solid red square with the letters 'a', 'b', and 'c' in black font at the top center.

a b c

A solid blue square with the letters 'a', 'b', and 'c' in black font at the top center.

a b c

a b c

```
insert(x, 1)
```

a x b c

a b c

```
insert(y, 2)
```

a b y c

a b c

insert(x, 1)

a x b c

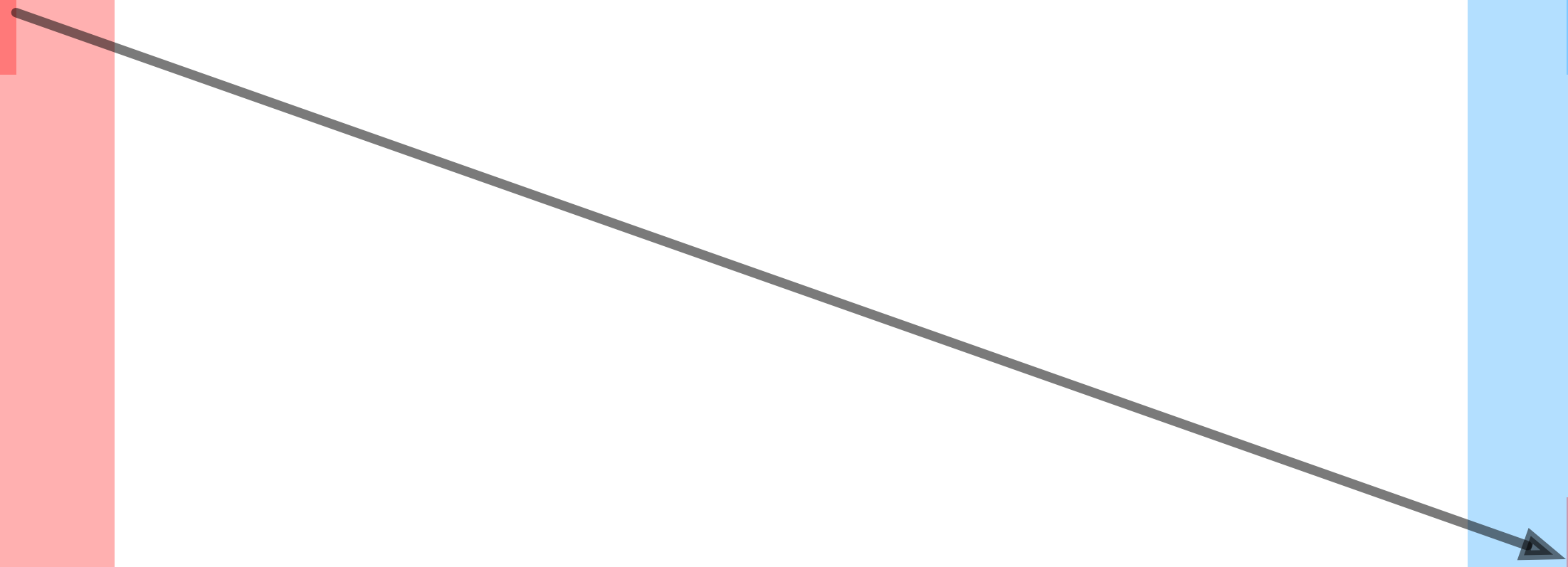
a b c

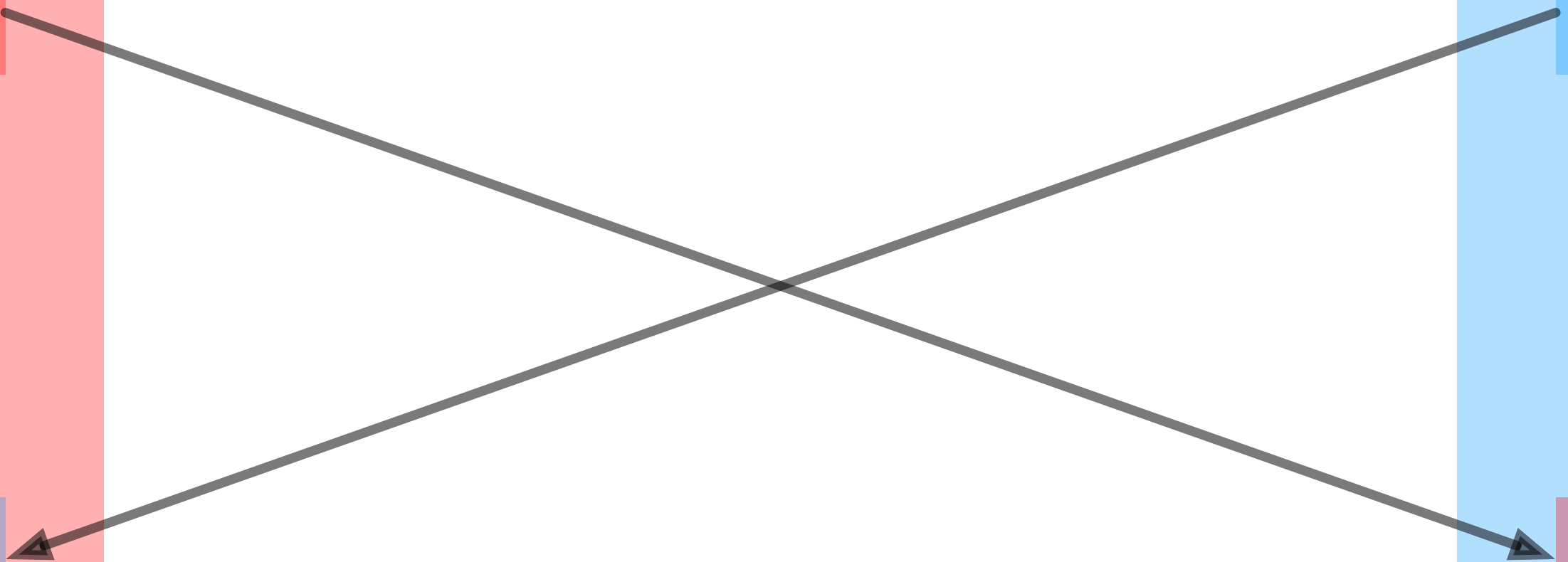
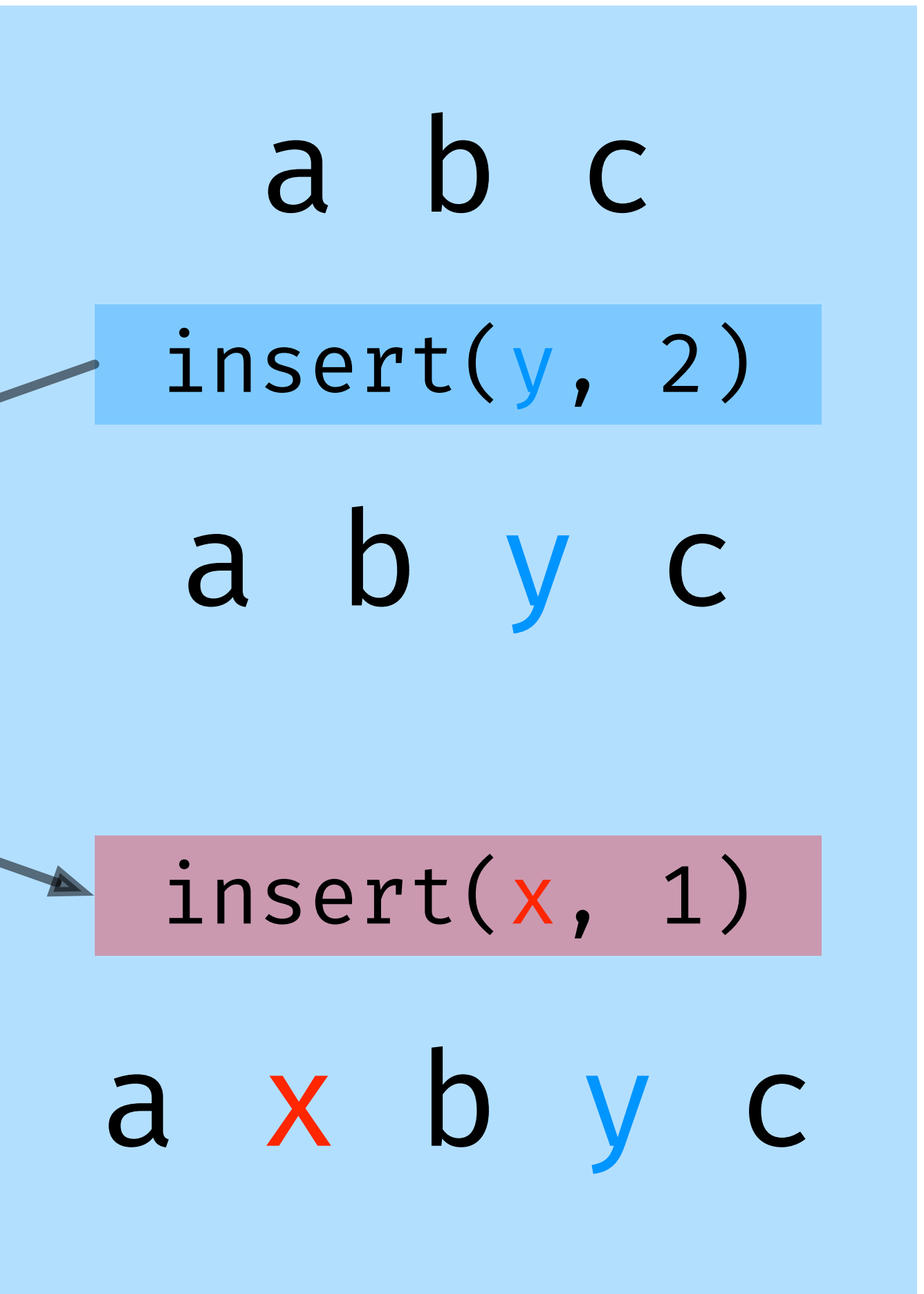
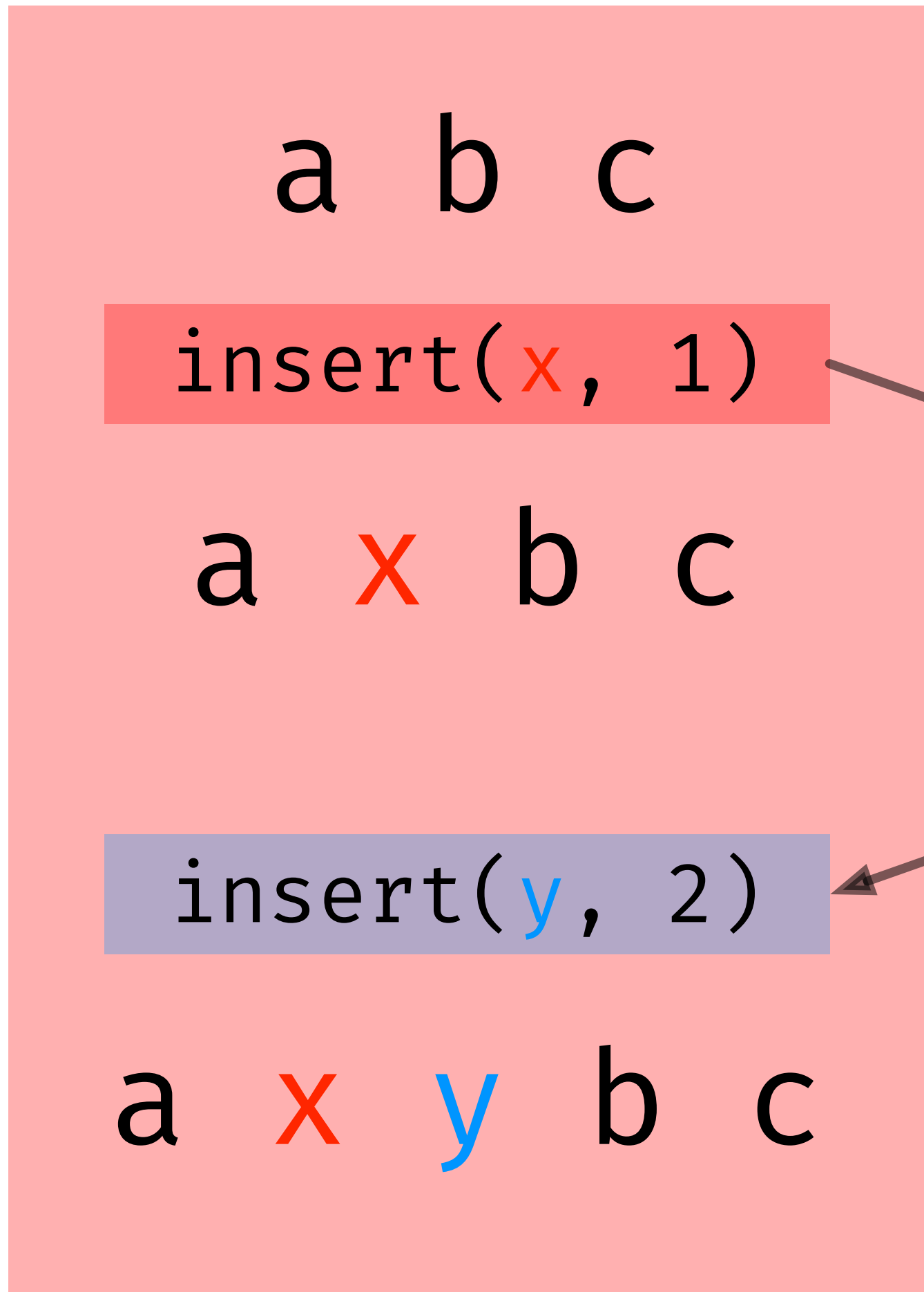
insert(y, 2)

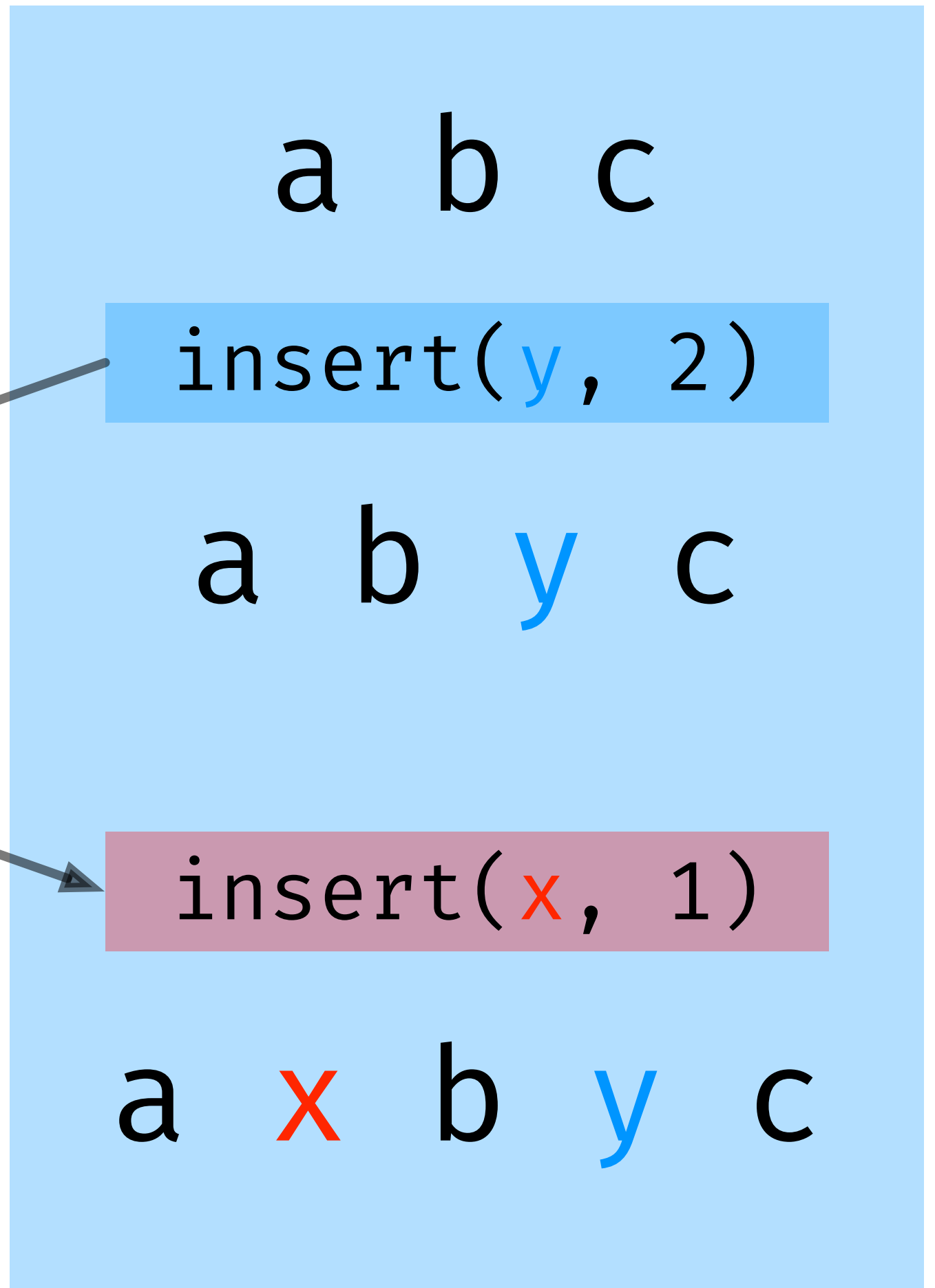
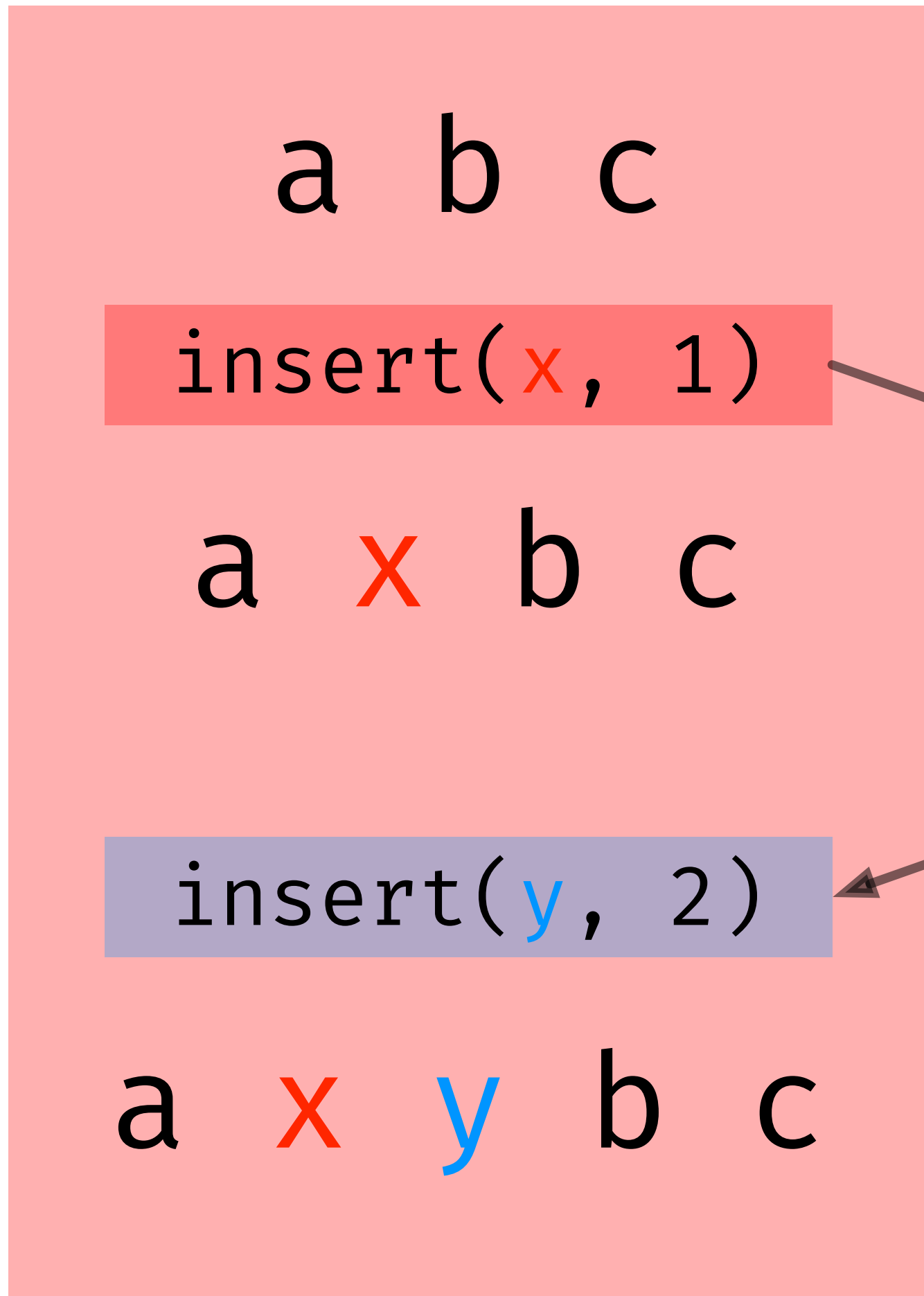
a b y c

insert(x, 1)

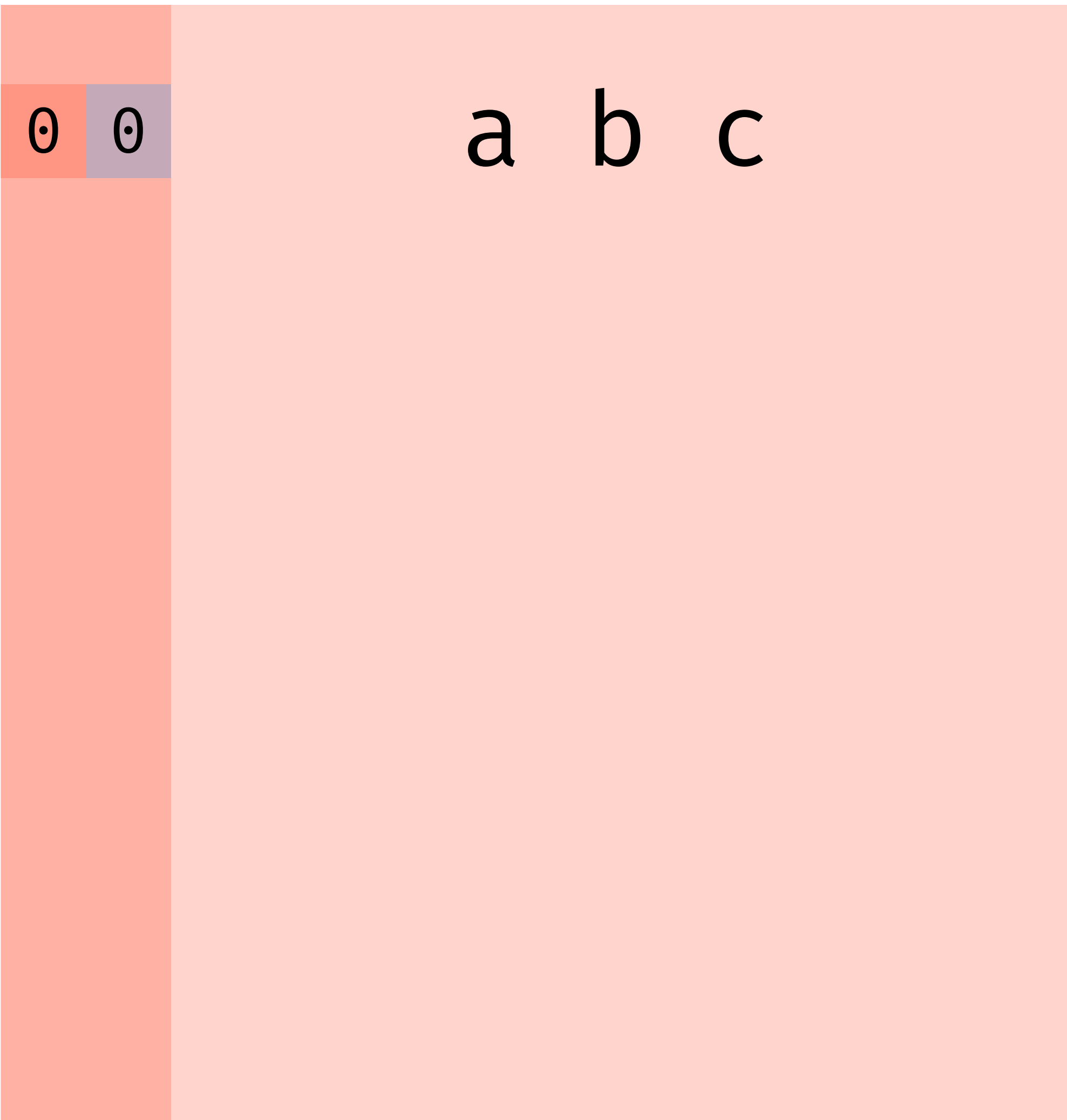
a x b y c







a x y b c 😭 a x b y c



0

0

a

b

c

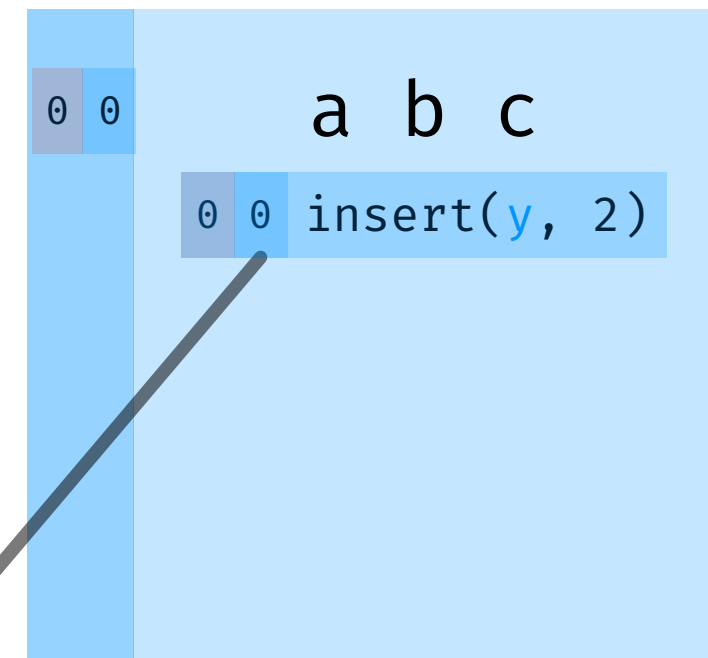
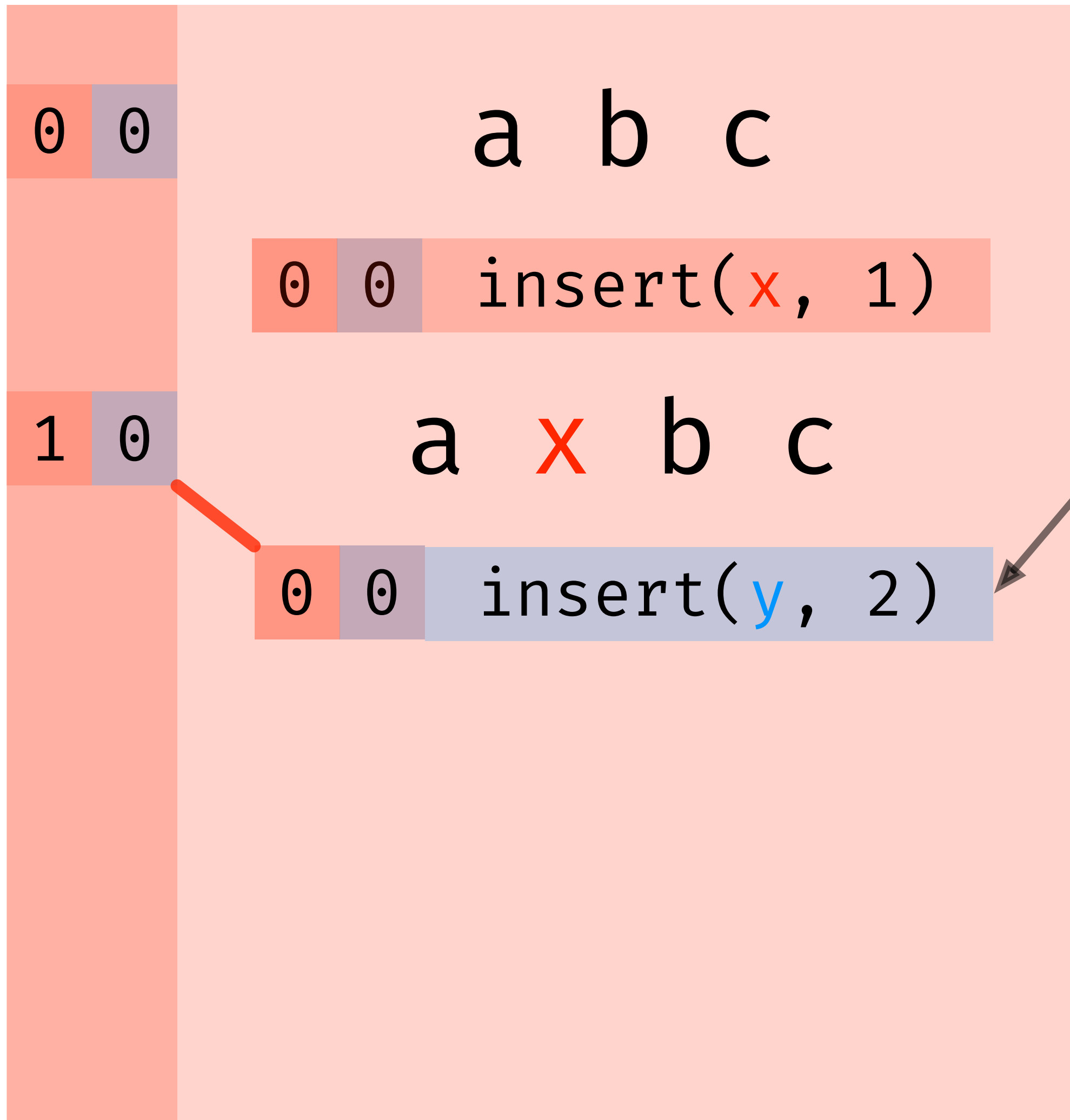
0 0

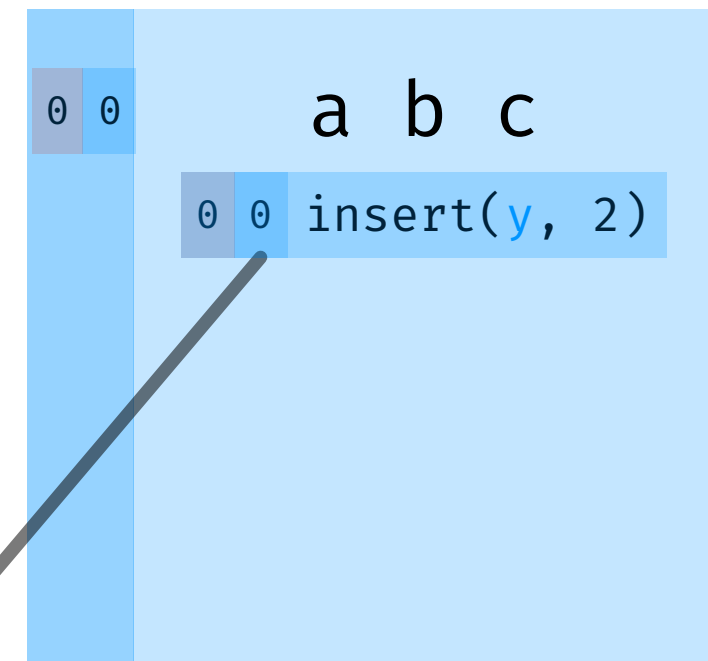
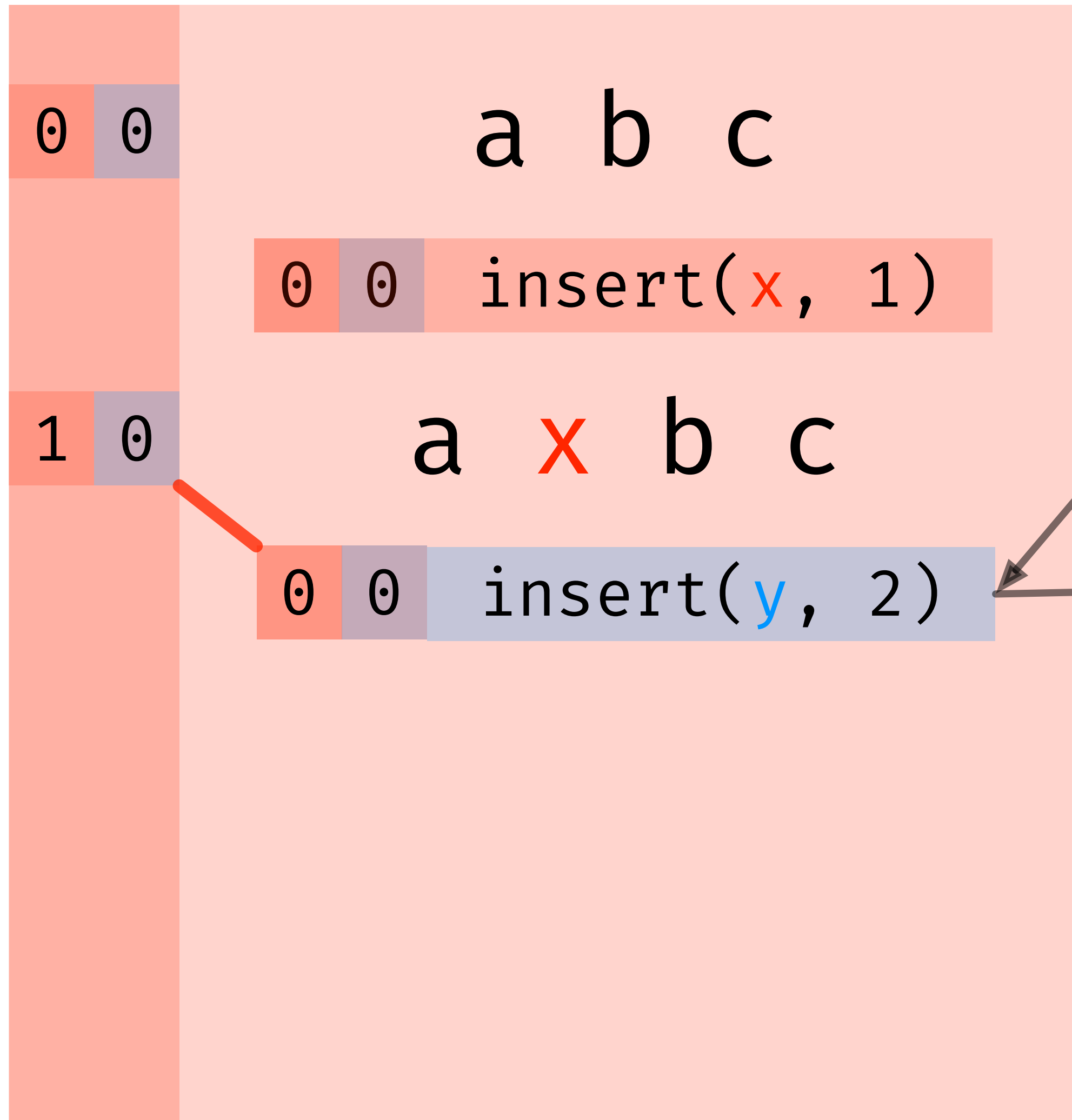
a b c

0 0 insert(x, 1)

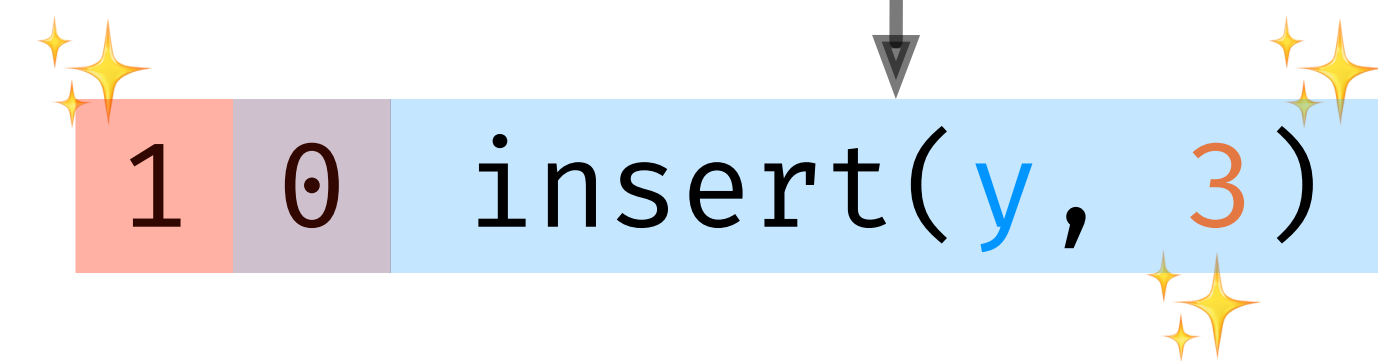
1 0

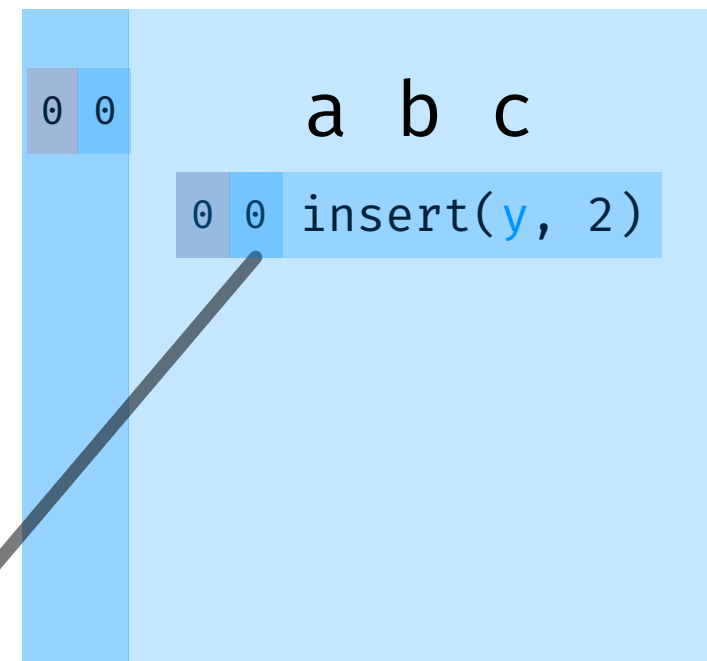
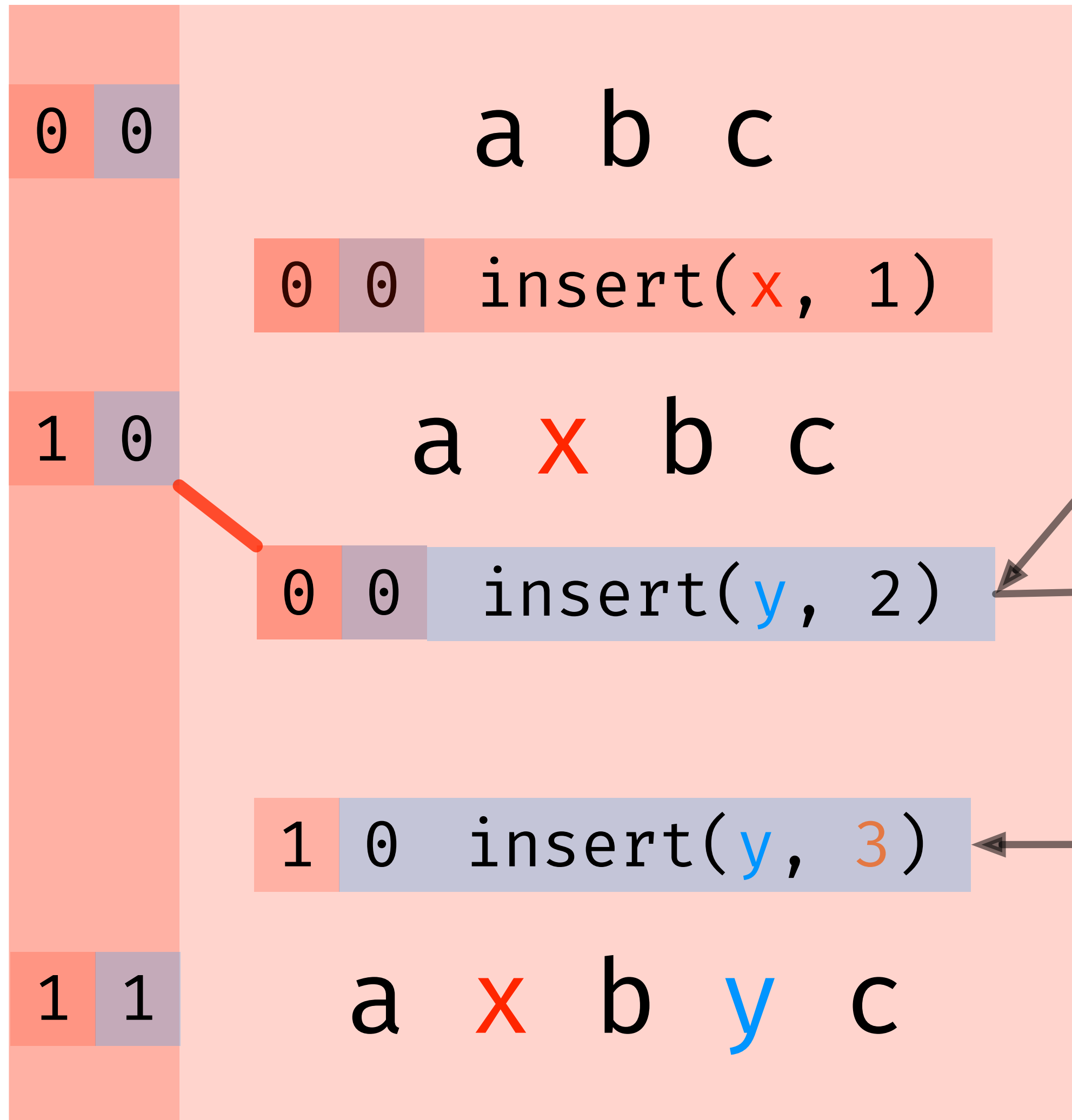
a x b c



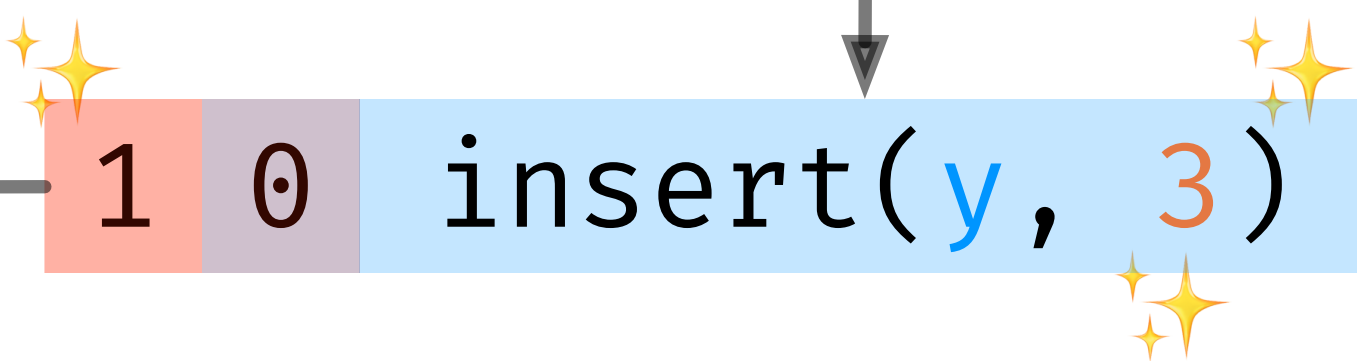


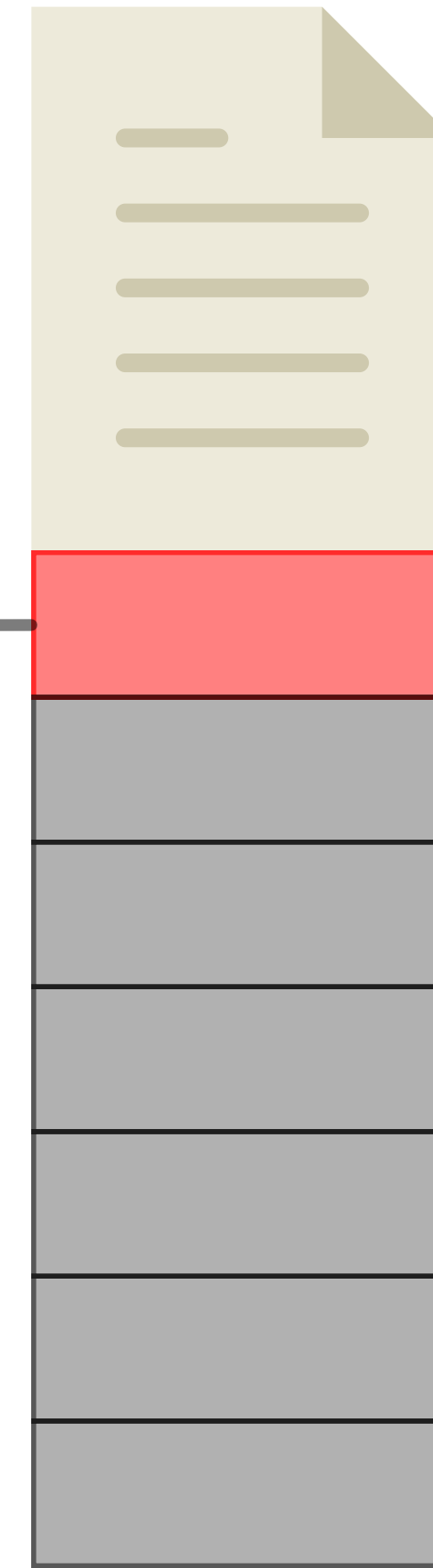
IT(insert(y, 2), insert(x, 1))





IT(insert(y, 2) , insert(x, 1))







Concurrent Undo Operations in Collaborative Editing Environments Using

Transparent Adaptation of Single-User Undo Operations for Multi-User Real-Time Collaborative Editing Systems

Tombstone Transformation Functions for Ensuring Consistency in Collaborative Editing Systems

Gérald Oster
Institute for Information Systems
ETH Zurich
Email: osterg@inf.ethz.ch

Abstract—In collaborative editing, consistency maintenance of the copies of shared data is a critical issue. In the past decade, Operational Transformation (OT) approaches have proposed a suitable mechanism for maintaining consistency. Unfortunately, none of the published propositions relying on this approach are able to satisfy the mandatory correctness properties TP_2 defined in the Ressel's framework. This paper addresses this correctness issue by proposing a new way to model the state by retaining tombstones when elements are reinserted. The instantiation of the proposed model for a linear data structure and the related transformation functions are provided.

I. INTRODUCTION

Collaborative editing systems allow users to edit a document from multiple sites across Internet. Depending on the work context, users can work synchronously or asynchronously. Synchronous collaboration is also called synchronous editing since when a user performs some modification on the document, these modifications are instantly visible to all users who can see them without any delay. In asynchronous collaboration, users may not work at the same time. They work in isolation: they can decide when to publish their modifications and when to integrate them into the document performed by other users.

In these systems, the shared documents are replicated at multiple sites. Parallel modifications on the same document may happen and therefore potential inconsistencies may occur. One of the main issues in collaborative editing is to maintain consistency of shared documents. Consistency maintenance mechanisms are classified in two categories depending on whether they are pessimistic or optimistic.

Pessimistic approaches try to give the impression that there is only one highly available copy in the whole system. In this copy – or part of a copy – can be edited at the same time by multiple users. This is achieved by

Operation Context and Context-based Operational Transformation

David Sun
Computer Science Division, EECS
University of California
Berkeley, CA
davidsun@cs.berkeley.edu

Chengzheng Sun
School of Computer Engineering
Nanyang Technological University
Singapore
CZSun@ntu.edu.sg

ABSTRACT

Operational Transformation (OT) is a technique for consistency maintenance in plain-text group editors [4]. In over 15 years, OT has evolved to support an increasing number of applications, including group undo [15, 19, 18, 21], group-awareness [28], operation notification and compression [20], spreadsheet and table-centric applications [14, 27], HTML/XML and tree-structured document editing [3, 7], word processing and slide creation [29, 25, 24], transparent and heterogeneous application-sharing [1, 10, 24], and mobile replicated computing and database systems [6, 16]. To effectively and efficiently support existing and new applications, we must continue to improve the capability and

1. INTRODUCTION

Operational Transformation (OT) was originally invented for consistency maintenance in plain-text group editors [4]. In over 15 years, OT has evolved to support an increasing number of applications, including group undo [15, 19, 18, 21], group-awareness [28], operation notification and compression [20], spreadsheet and table-centric applications [14, 27], HTML/XML and tree-structured document editing [3, 7], word processing and slide creation [29, 25, 24], transparent and heterogeneous application-sharing [1, 10, 24], and mobile replicated computing and database systems [6, 16]. To effectively and efficiently support existing and new applications, we must continue to improve the capability and

Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems

CHENGZHENG SUN
Griffith University
XIAOHUA JIA
City University of Hong Kong
YANCHUN ZHANG
The University of Southern Queensland
YUN YANG
Deakin University
and
DAVID CHEN
Griffith University

Real-time cooperative editing systems allow multiple users to view and edit the same text/graphic/image/multimedia document at the same time from multiple sites connected by communication networks. Consistency maintenance is one of the most significant challenges in designing and implementing real-time cooperative editing systems. In this article, a consistency model, with properties of convergence, causality preservation, and intention preservation, is proposed as a framework for consistency maintenance in real-time cooperative editing systems. Moreover, an integrated set of schemes and algorithms, which support the proposed consistency model, are devised and discussed in detail. In particular, we have contributed (1) a novel generic operation transformation control algorithm for achieving convergence and causality preservation in combination with schemes for intention preservation

Undo and Exclusion in Collaborative Editing Systems

Xiaohua Jia
Department of Computer Science
City University of Hong Kong
Kowloon, Hong Kong
jia@cs.cuhk.edu.hk

Undo as Concurrent Inverse in Group Editors

CHENGZHENG SUN
Griffith University

An important mechanism for error recovery and exploration of alternatives in interactive and collaborative applications, an undo facility should have the capability of undoing any operation at any time. However, supporting undo in collaborative applications is technically challenging and one of the existing group undo solutions is able to offer such a capability. In this article, we propose a novel theory of undo in collaborative editing systems called Concurrent Inverse Operational Transformation (CIOT)—which has been implemented in a group editor. CIOT is potentially applicable to distributed applications.

Undo as Concurrent Inverse in Group Editors

CHENGZHENG SUN
Griffith University
XIAOHUA JIA
City University of Hong Kong
YANCHUN ZHANG
The University of Southern Queensland
YUN YANG
Deakin University
and
DAVID CHEN
Griffith University

1. INTRODUCTION

Undo is a key feature in interactive applications. Many familiar single-user applications, including text editors, word processors, design tools and even Web browsers, allow the user to undo operations in a chronological order. Undo is often used for user-level error recovery, e.g., to fix typos. It can also encourage users to explore unfamiliar capabilities in an application provided that the effects of erroneous operations can always be undone.

In collaborative applications, undo is significantly more challenging [9, 2, 14]. When the users are distributed and work in parallel, local and remote operations could be interleaved arbitrarily due to concurrency. A user request to undo could mean to undo a local operation or a remote operation. To undo a remote operation is even trickier because there can be multiple operations by several remote users and it is important to unambiguously specify which operation by which user. Otherwise, the undo action could be interpreted arbitrarily at different sites, leading to unpredictable consequences in the system. Therefore, in a distributed environment, it becomes necessary to support selective undo operations. In this paper, we propose a selective undo algorithm level, i.e., to undo the effects of any selected

Operational Transformation for Collaborative Editing Systems

CHENGZHENG SUN
Griffith University

collaborative editing systems. In this article, we propose a novel theory of Operational Transformation (OT)—which has been implemented in a group editor. OT is potentially applicable to distributed applications.

collaborative editing systems. In this article, we propose a novel theory of Operational Transformation (OT)—which has been implemented in a group editor. OT is potentially applicable to distributed applications.

collaborative editing systems. In this article, we propose a novel theory of Operational Transformation (OT)—which has been implemented in a group editor. OT is potentially applicable to distributed applications.

collaborative editing systems. In this article, we propose a novel theory of Operational Transformation (OT)—which has been implemented in a group editor. OT is potentially applicable to distributed applications.

collaborative editing systems. In this article, we propose a novel theory of Operational Transformation (OT)—which has been implemented in a group editor. OT is potentially applicable to distributed applications.

collaborative editing systems. In this article, we propose a novel theory of Operational Transformation (OT)—which has been implemented in a group editor. OT is potentially applicable to distributed applications.

collaborative editing systems. In this article, we propose a novel theory of Operational Transformation (OT)—which has been implemented in a group editor. OT is potentially applicable to distributed applications.

collaborative editing systems. In this article, we propose a novel theory of Operational Transformation (OT)—which has been implemented in a group editor. OT is potentially applicable to distributed applications.

Concurrent Undo Operations in Collaborative Editing Environments Using

Transparent Adaptation of Single-User Undo Operations for Multi-User Real-Time

Operation Context and Context-based Operational Transformation

David Sun
Computer Science Division, EECS
University of California
Berkeley, CA
davidsun@cs.berkeley.edu

Chengzheng Sun
School of Computer Engineering
Nanyang Technological University
Singapore
CZSun@ntu.edu.sg

ABSTRACT

Operational Transformation (OT) is a technique for consistency maintenance in plain-text group editors [4]. In over 15 years, OT has evolved to support an increasing number of applications, including group undo [15, 19, 18, 21], group-awareness [28], operation notification and compression [20], spreadsheet and table-centric applications [14, 27], HTML/XML and tree-structured document editing [3, 7], word processing and slide creation [29, 25, 24], transparent and heterogeneous applications [1, 10, 26], and mobile real-time collaborative editing [11, 10, 26], and new problems. The theory of causality has been crucial in determining its correctness requirements. Past approaches to work around this problem have implemented ad-hoc algorithms, but it is inadequate to work around this problem. In this article, we propose a novel theory of causality, which has been implemented in a distributed applications.

1. INTRODUCTION

Operational Transformation (OT) was originally invented for consistency maintenance in plain-text group editors [4]. In over 15 years, OT has evolved to support an increasing number of applications, including group undo [15, 19, 18, 21], group-awareness [28], operation notification and compression [20], spreadsheet and table-centric applications [14, 27], HTML/XML and tree-structured document editing [3, 7], word processing and slide creation [29, 25, 24], transparent and heterogeneous applications [1, 10, 26], and mobile real-time collaborative editing [11, 10, 26], and new problems. The theory of causality has been crucial in determining its correctness requirements. Past approaches to work around this problem have implemented ad-hoc algorithms, but it is inadequate to work around this problem. In this article, we propose a novel theory of causality, which has been implemented in a distributed applications.

tion and exclusion
ing-wise operations in
liting systems

Xiaohua Jia
Department of Computer Science
City University of Hong Kong
Kowloon, Hong Kong
jia@cs.cuhk.edu.hk

ONS ON PARALLEL AND DISTRIBUTED SYSTEMS,

Operational Transformation for Collaborative Editing Systems

number, IEEE

collaborative editing systems
collaboration over communication networks
to meet essential OT requirements
propose a novel theory of causality
based OT (COT)—which provides a
m has been implemented in a distributed
are potentially applicable to

Tombstone Transformation Functions for Ensuring Consistency in Collaborative Editing Systems

Gérald Oster
Institute for Information Systems
ETH Zurich
Email: osterg@inf.ethz.ch

Abstract—In collaborative editing, consistency maintenance of the copies of shared data is a critical issue. In the past decade, Operational Transformation (OT) approaches have proposed a suitable mechanism for maintaining consistency. Unfortunately, none of the published propositions relying on this approach are able to satisfy the mandatory correctness properties TP_2 defined in the Ressel's framework. This paper addresses this correctness issue by proposing a new way to model consistency by retaining tombstones when elements are re-instantiated of the proposed model for a linear data structure and the related transformation functions are provided.

I. INTRODUCTION

Collaborative editing systems allow users to edit a document from multiple sites across Internet. Depending on the work context, users can work synchronously or asynchronously. Synchronous collaboration is also called synchronous editing since when a user performs some modification to the document, these modifications are instantly visible to all users who can see them without any delay. In asynchronous collaboration, users may not work at the same time. They work in isolation: they can decide when to publish their modifications and when to integrate them with the modifications performed by other users.

In these systems, the shared documents are replicated at multiple sites. Parallel modifications to the same document may happen and therefore potential inconsistencies may occur. One of the main issues in collaborative editing is to maintain consistency of shared documents. Consistency maintenance mechanisms are classified in two categories depending on whether they are pessimistic or optimistic.

Pessimistic approaches try to give the impression that there is only one highly available copy in the whole system. In this copy – or part of a copy – can be edited at the same time by all users. This is the case of the

Decentralized Multi-User Undo String-Wise Operations Consistently Low Overhead

Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems

CHENGZHENG SUN
Griffith University
XIAOHUA JIA
City University of Hong Kong
YANCHUN ZHANG
The University of Southern Queensland
YUN YANG
Deakin University
and
DAVID CHEN
Griffith University

Real-time cooperative editing systems allow multiple users to view and edit the same text/graphic/image/multimedia document at the same time from multiple sites connected by communication networks. Consistency maintenance is one of the most significant challenges in designing and implementing real-time cooperative editing systems. In this article, a consistency model, with properties of convergence, causality preservation, and intention preservation, is proposed as a framework for consistency maintenance in real-time cooperative editing systems. Moreover, an integrated set of schemes and algorithms, which support the proposed consistency model, are devised and discussed in detail. In particular, we have contributed (1) a novel generic operation transformation control algorithm for achieving intention preservation in combination with schemes for achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems.

1. INTRODUCTION

Undo is a key feature in interactive applications. Many familiar single-user applications, including text editors, word processors, design tools and even Web browsers, allow the user to undo operations in a chronological order. Undo is often used for user-level error recovery, e.g., to fix typos. It can also encourage users to explore unfamiliar capabilities in an application provided that the effects of erroneous operations can always be undone. In collaborative applications, undo is significantly more challenging [9, 2, 14]. When the users are distributed and work in parallel, local and remote operations could be interleaved arbitrarily due to concurrency. A user request to undo could mean to undo a local operation or a remote operation. To undo a remote operation is even trickier because there can be multiple operations by several remote users and it is important to unambiguously specify which operation by which user. Otherwise, the undo action could be interpreted arbitrarily at different sites, leading to unpredictable consequences in the system. Therefore, in a distributed environment, it becomes necessary to support selective undo operations at the algorithm level, i.e., to undo the effects of any selected operations.

1. INTRODUCTION

Undo is a key feature in interactive applications. Many familiar single-user applications, including text editors, word processors, design tools and even Web browsers, allow the user to undo operations in a chronological order. Undo is often used for user-level error recovery, e.g., to fix typos. It can also encourage users to explore unfamiliar capabilities in an application provided that the effects of erroneous operations can always be undone. In collaborative applications, undo is significantly more challenging [9, 2, 14]. When the users are distributed and work in parallel, local and remote operations could be interleaved arbitrarily due to concurrency. A user request to undo could mean to undo a local operation or a remote operation. To undo a remote operation is even trickier because there can be multiple operations by several remote users and it is important to unambiguously specify which operation by which user. Otherwise, the undo action could be interpreted arbitrarily at different sites, leading to unpredictable consequences in the system. Therefore, in a distributed environment, it becomes necessary to support selective undo operations at the algorithm level, i.e., to undo the effects of any selected operations.

Undo as Concurrent Inverse in Group Editors

CHENGZHENG SUN
Griffith University

an important mechanism for error recovery and exploration of alternatives in interactive and collaborative applications, an undo facility should have the capability of undoing any operation at any time. However, supporting undo in collaborative applications is technically challenging and time-consuming. The existing group undo solutions are able to offer such a capability. In this article, we propose a novel theory of causality, which has been implemented in a distributed applications.

an important mechanism for error recovery and exploration of alternatives in interactive and collaborative applications, an undo facility should have the capability of undoing any operation at any time. However, supporting undo in collaborative applications is technically challenging and time-consuming. The existing group undo solutions are able to offer such a capability. In this article, we propose a novel theory of causality, which has been implemented in a distributed applications.

Keywords: Distributed Editing, Undo, Consistency, Causality, Intention Preservation, and Convergence.

Authors: Chengzheng Sun, Xiaohua Jia, Yanchun Zhang, Yun Yang, David Chen.

Address: Chengzheng Sun, Griffith University, St. Lawrence, QLD, Australia.

Address: Xiaohua Jia, City University of Hong Kong, Kowloon, Hong Kong.

Address: Yanchun Zhang, The University of Southern Queensland, St. Lawrence, QLD, Australia.

Address: Yun Yang, Deakin University, Geelong, VIC, Australia.

Address: David Chen, Griffith University, St. Lawrence, QLD, Australia.

Address: Gerald Oster, Institute for Information Systems, ETH Zurich, Zurich, Switzerland.

applications
use undo to

Grant No.

University,

room use is

commercial

office is given

servers, or to

pages 309–361.

distributed applications

ential OT-required

ns of other types, i

ons that are gener

the limitations have

exity problems.

nd caused corre

OT history. The c

pair of operation

ns are concurr

t the concurr

another essentia

involved in a

ame document

not be defined

n OT system

h other. The

ure relevant

well-known

olved by

causality

minent

m in

l for

ing

T-

k

Concurrent Undo Operations in
Environments Using

Operation Context and Context-based Operational Transformation

David Sun
Computer Science Division, EECS
University of California
Berkeley, CA
davidsun@cs.berkeley.edu

Chengzheng Sun
School of Computer Engineering
Nanyang Technological University
Singapore
CZSun@ntu.edu.sg

and exclusion
ing-wise operations in
liting systems

Xiaohua Jia
Department of Computer Science
City University of Hong Kong
Kowloon, Hong Kong
jia@cs.cuhk.edu.hk

ONS ON PARALLEL AND DISTRIBUTED SYSTEMS,
rational Transform

Editing Sys

ber, IEEE

collaborative editing systems
collaboration over communi
te to meet essential OT r
propose a novel theory
ed OT (COT)—which pro
m has been implemente
are potentially applicab

distributed applications

ential OT-required
ns of other types, i
ons that are gener
e limitations have
exity problems.
d caused corre
OT history. The c
pair of concurr
ns are concurr
t the concurr
another essentia
involved in a
ame document
n OT system
h other. The
re relevant
ell-known
olved by
causality

ork: Distributed
Tools and Tech
ms—human fac
ory and methods;

ce maintenance,
rted cooperative

Grant No.
h University.

room use is
commercial
vice is given
ervers, or to

es 309–361.
minent
m in
l for
ing
T-
k

"Unfortunately, implementing OT sucks. There's a million algorithms with different tradeoffs, mostly trapped in academic papers. The algorithms are really hard and time consuming to implement correctly. Wave took two years to write and if we rewrote it today, it would take almost as long to write a second time."

– Google Wave Engineer

Tombs
Consistency in Col
Gérald O
Institute for Information System
ETH Zurich
Email: osterg@i

Abstract—In collaborative editing, consistency of the copies of shared data is a critical issue. Over the decade, Operational Transformation (OT) has evolved to support an increasing number of applications, including group undo [1, 9, 18, 21], group-awareness [28], operation notification [29], and mobile replicated computing and database systems [6, 16]. To design and efficiently support existing and new OT algorithms, we reflected on the theory of OT and its practical applications. In this article, we propose a novel theory of OT (COT)—which has been implemented in Google Wave—and discuss its practical applications. The idea is to interpret OT as a sequence of operations that are potentially applicable in any order. This approach allows us to design a novel theory of OT (COT) which is potentially applicable in any order. This approach allows us to design a novel theory of OT (COT) which is potentially applicable in any order.

Collaborative editing systems allow users to edit a document from multiple sites across Internet. Despite the work context, users can work synchronously or asynchronously. Synchronous collaboration is also called editing since when a user performs some modification to the document, these modifications are instantly visible to users who can see them without any delay. In asynchronous collaboration, users may not see the same time. They work in isolation; they can decide when to publish their modifications and when to integrate them with the modifications performed by other users.

In these systems, the shared documents are replicated at multiple sites. Parallel modifications to the same document may happen and therefore potential inconsistencies may occur. One of the main issues in collaborative editing is to maintain consistency of shared documents. Consistency maintenance mechanisms are classified into two categories depending on whether they are pessimistic or optimistic.

Pessimistic approaches try to give the impression that only one highly available copy in the whole system is the authoritative copy. This copy can be edited at the same time by multiple users. This approach is based on the assumption that the system will eventually converge to a single state.

Real-time cooperative editing systems allow multiple users to view and edit the same text/graphic/image/multimedia document at the same time from multiple sites connected by communication networks. Consistency maintenance is one of the most significant challenges in designing and implementing real-time cooperative editing systems. In this article, a consistency model, with properties of convergence, causality preservation, and intention preservation, is proposed as a framework for consistency maintenance in real-time cooperative editing systems. Moreover, an integrated set of schemes and algorithms, which support the proposed consistency model, are devised and discussed in detail. In particular, we have contributed (1) a novel generic operation transformation control algorithm for achieving intention preservation in combination with schemes for achieving convergence, causality preservation, and intention preservation.

A light red rectangular box with a thin red border. Inside the box, the lowercase letters 'a', 'b', and 'c' are arranged horizontally in the upper portion of the box.

a b c

A light blue rectangular box with a thin blue border. Inside the box, the lowercase letters 'a', 'b', and 'c' are arranged horizontally in the upper portion of the box.

a b c

a b c

```
insert(x, ✨)
```

a x b c

a b c

a b c

```
insert(x, ✨)
```

a x b c

a b c

```
insert(y, 🦄)
```

a b y c

a b c

insert(x, ✨)

a x b c

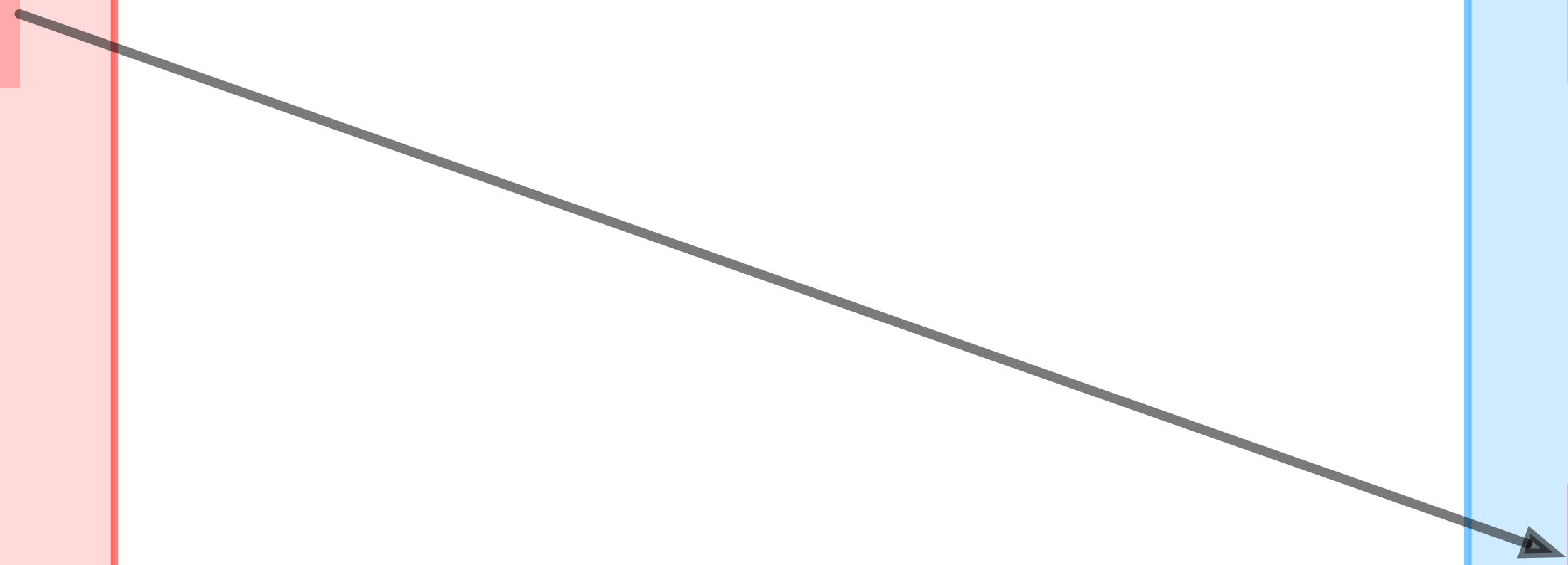
a b c

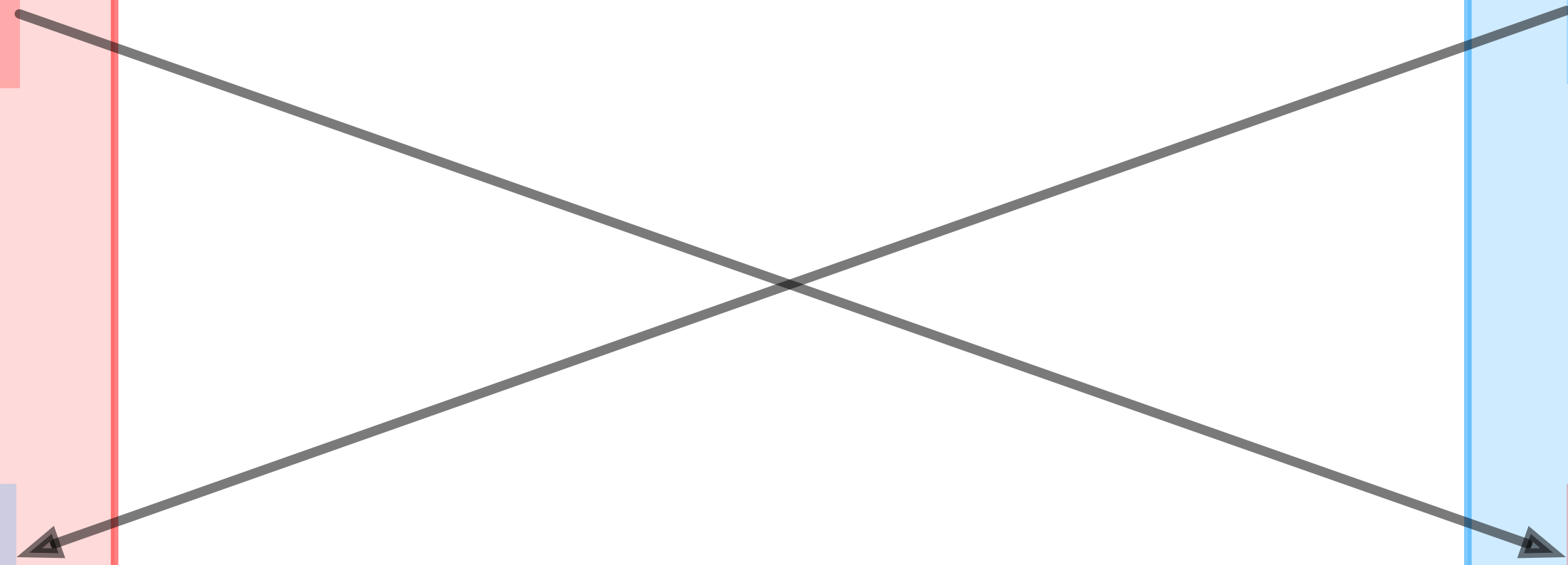
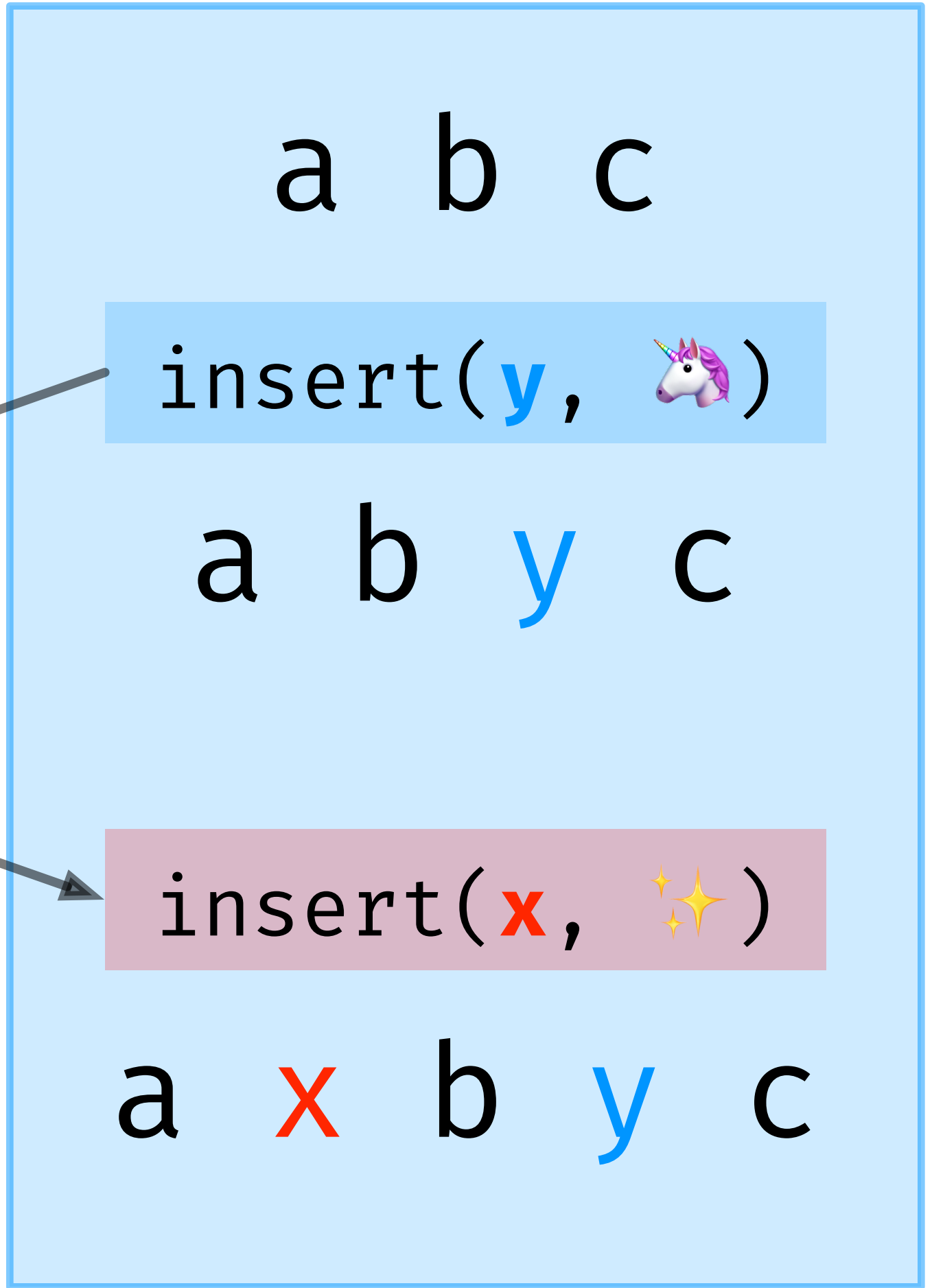
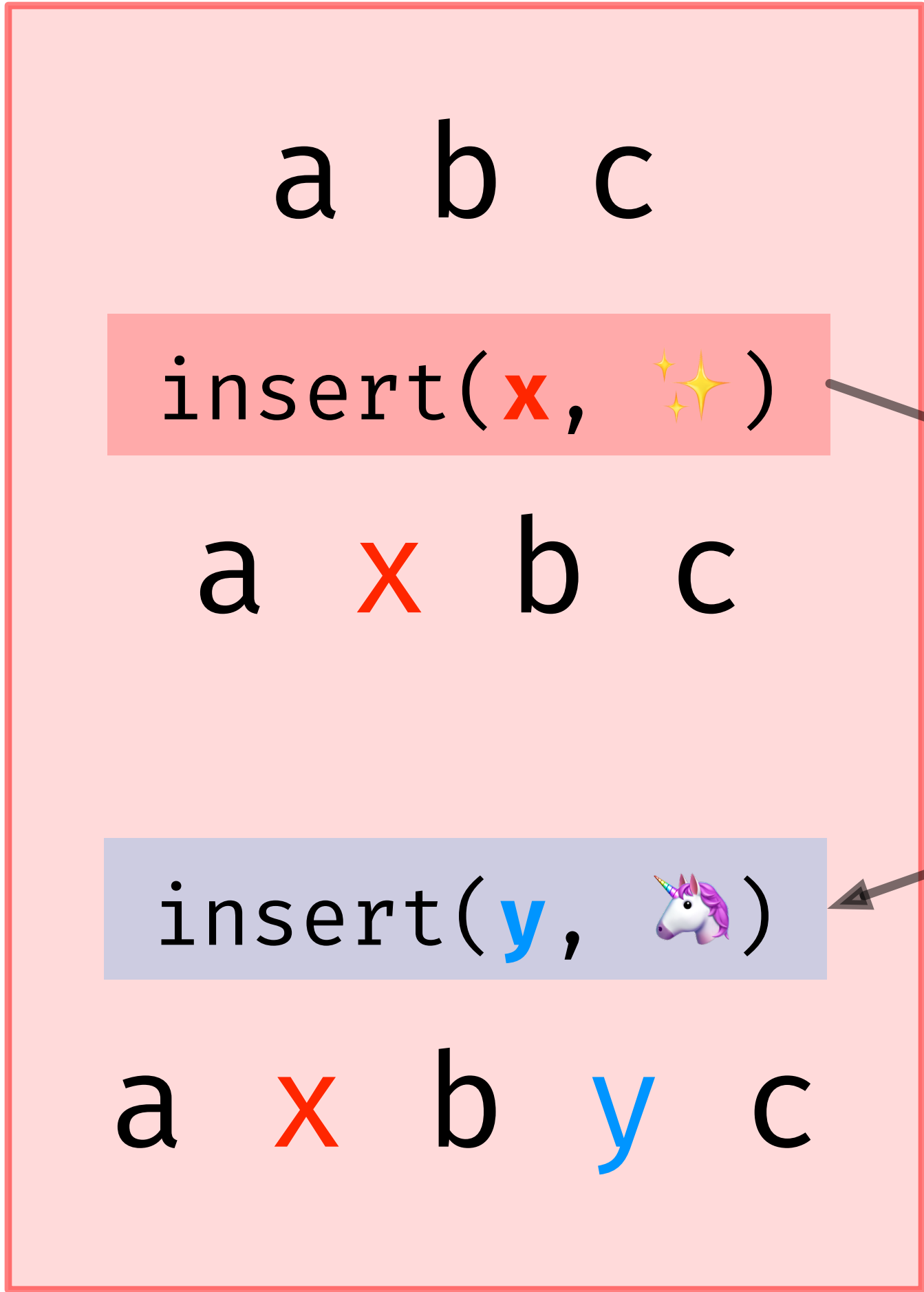
insert(y, 🦄)

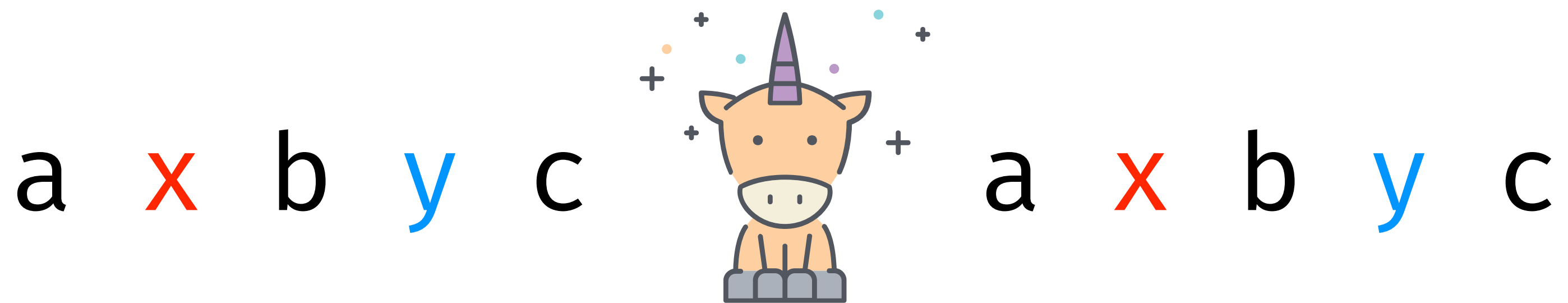
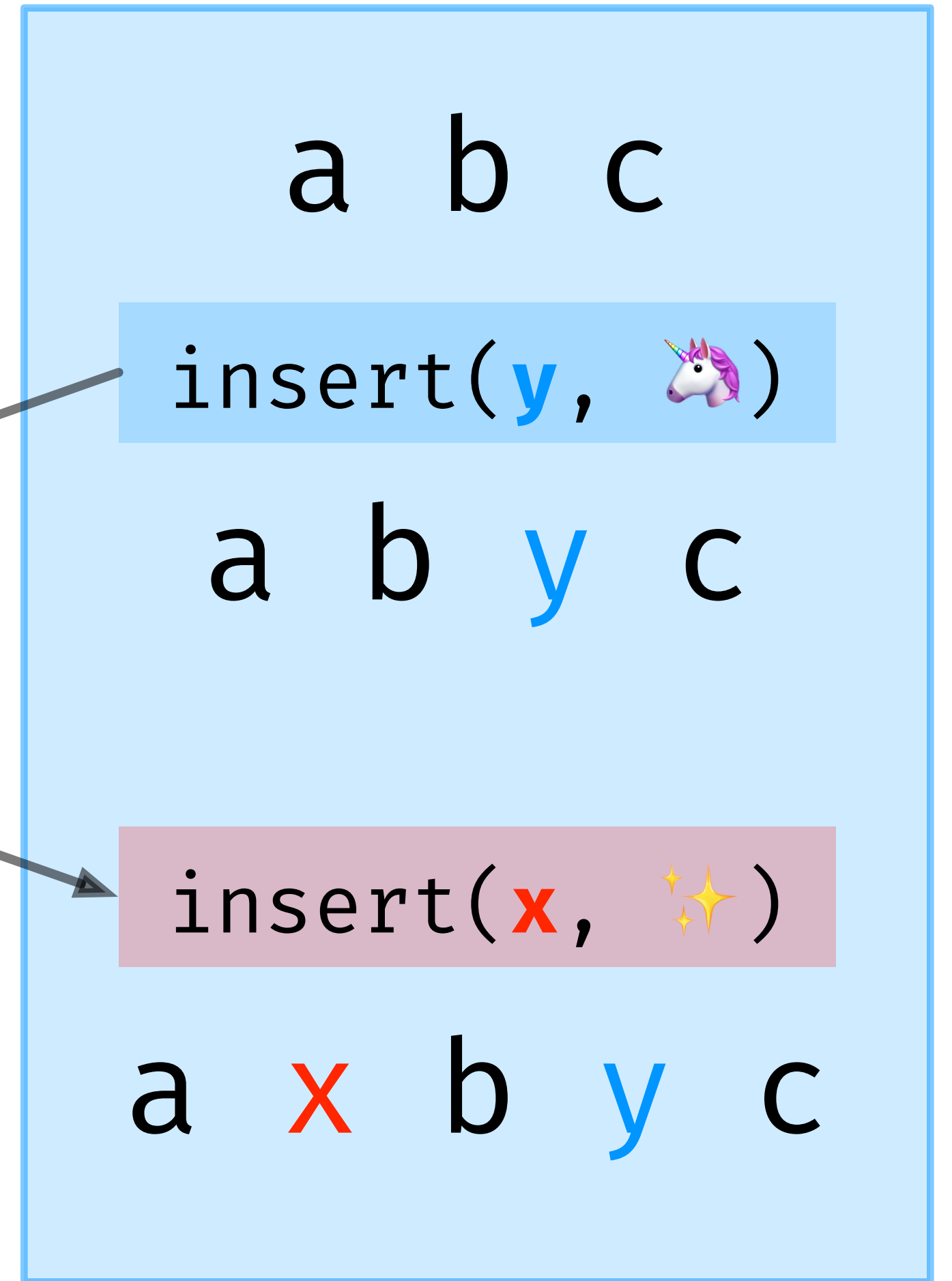
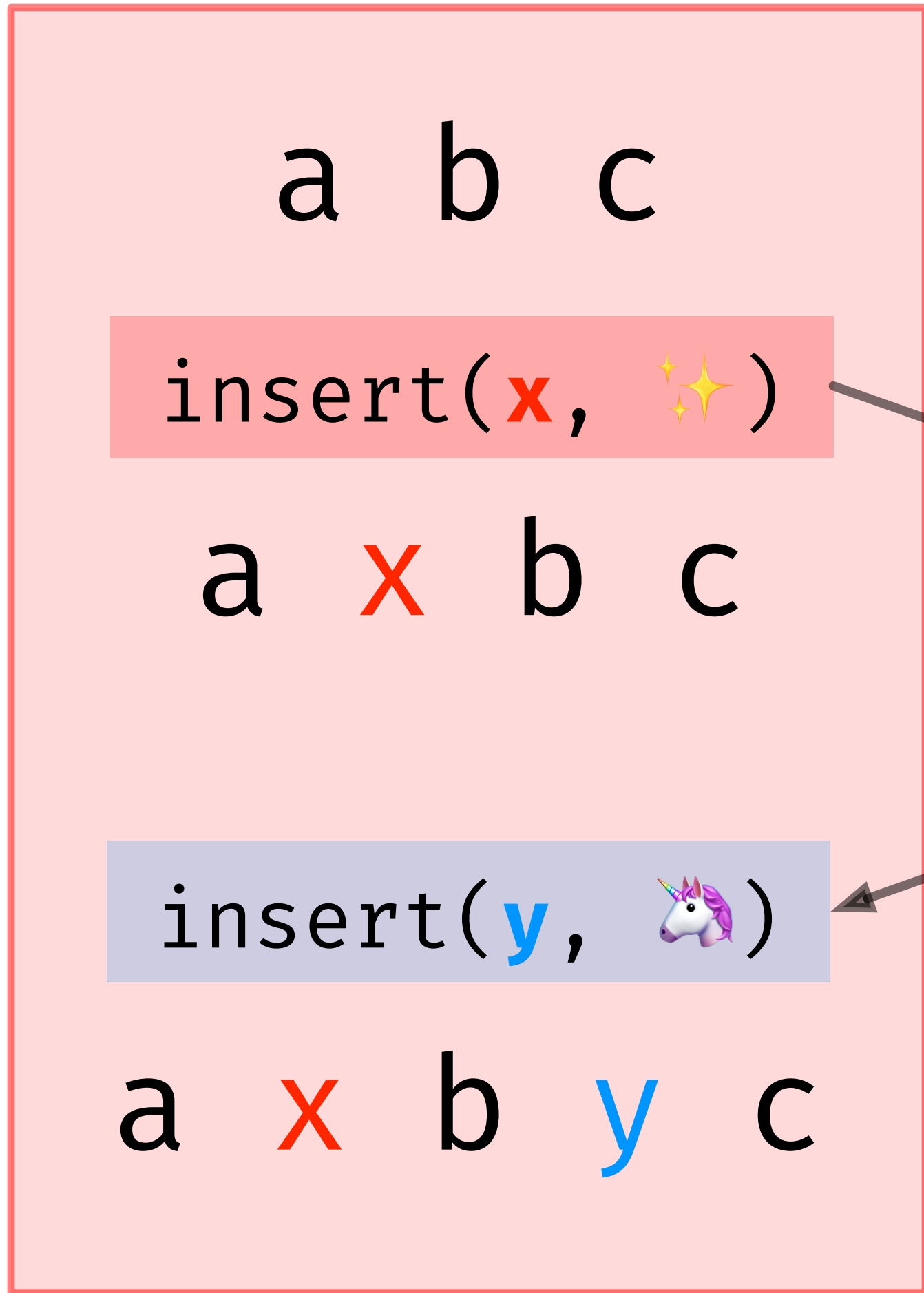
a b y c

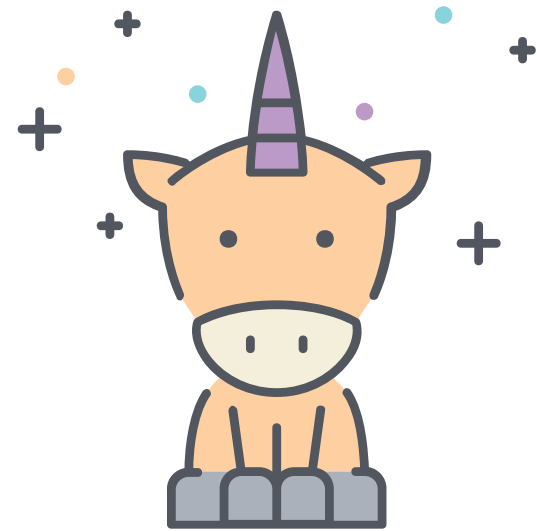
insert(x, ✨)

a x b y c









CRDTs

Conflict-Free Replicated Data Types

Data Consistency for P2P Collaborative Editing

Gérald Oster
Institute for Information Systems
ETH Zurich
osterger@inf.ethz.ch

Pascal Urso, Pascal Molli, Abdessamad Imine
Université Henri Poincaré, Nancy 1
LORIA
{urso,molli,imine}@loria.fr

ABSTRACT

Peer-to-peer (P2P) networks are very efficient for distributing content. We want to use this potential to allow not only distribution but collaborative editing of this content. Existing collaborative editing systems are centralised or depend on the number of sites. Such systems cannot scale when deployed on P2P networks. In this paper, we propose a new model for building a collaborative editing system. This model is fully decentralised and does not depend on the number of sites.

Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed Systems—*Distributed applications*; H.5.3 [Information Interfaces and Presentation]: Group and Organization Interfaces—*Collaborative computing, theory and models*

General Terms

Algorithms, Design, Human Factors

Keywords

CSCW, Collaborative editing, Optimistic replication, Concurrency control

1. INTRODUCTION

Currently, peer-to-peer systems demonstrated how they can ensure scalable content distribution. In their survey [4], Androutsellis-Theotokis et al. wrote:

“Peer-to-peer content distribution systems rely on the replication of content on more than one node for improving the availability of content, enhancing performance and resisting censorship attempts.”

We want to reuse these characteristics not only for content distribution but also for content editing. Currently, P2P

networks mainly distribute immutable contents, we want to distribute updates on this content and manage collaborative editing on it. We are convinced that if we can deploy a group editor framework on a P2P network, we open the way for P2P content editing. It means that all existing collaborative editing applications such as CVS and Wiki can be redeployed on P2P networks and take advantage of availability improvements, performance enhancements and censorship resistance of P2P networks. For instance, Wikipedia [2] is currently a collaborative encyclopædia that has collected more than 4,700,000 articles in more than 200 languages. Wikipedia has more than 50 million of page requests per day. 200,000 changes are made every day [1]. However, Wikipedia needs a costly infrastructure to handle the load. Hundreds of thousands of dollars are spent every year to fund the infrastructure. A P2P massive collaborative editing system would allow to distribute the service, tolerate failures, improve performances, resist to censorship and share the cost of the underlying infrastructure.

Collaborative editing systems such as CVS or Wikis are currently centralised and cannot be adapted to peer-to-peer networks. Collaborative systems based on the operational transformation approach [7, 23] can be decentralised.

However, existing algorithms such as GOTO [23], ABT[16] and SOCT2 [21] rely on vector clocks to detect concurrent operations. OT approach supposes that each operation is immediately executed locally, stored in a local log and then broadcast to other sites in order to be re-executed and stored in their logs. A vector clocks is associated to each operation. A vector clocks [17] is an array of logical clocks, one clock per site. It is used to detect the happened-before relationship and therefore the concurrency between operation. It causes no problem if the number of sites is fixed and low but if the number of sites grows, the size of the vector clocks is unbounded. Thus, messages exchanged between sites will grow as well as the size of local operation. Also, the time efficiency of operation on vectors clocks will decline as vectors clocks grow. Clearly, vectors clocks prevent these algorithms to scale and represent a serious bottleneck for their deployment on P2P networks.

In this paper we propose a new model called WOOT for building group editors that is suitable for dynamic P2P systems. Compared to existing decentralised group editor models, the number of sites involved in group editing is not a variable.

The remainder of this paper is organised as follows: Section 2 describes the WOOT approach and details the consistency model. Section 3 presents a formal definition of

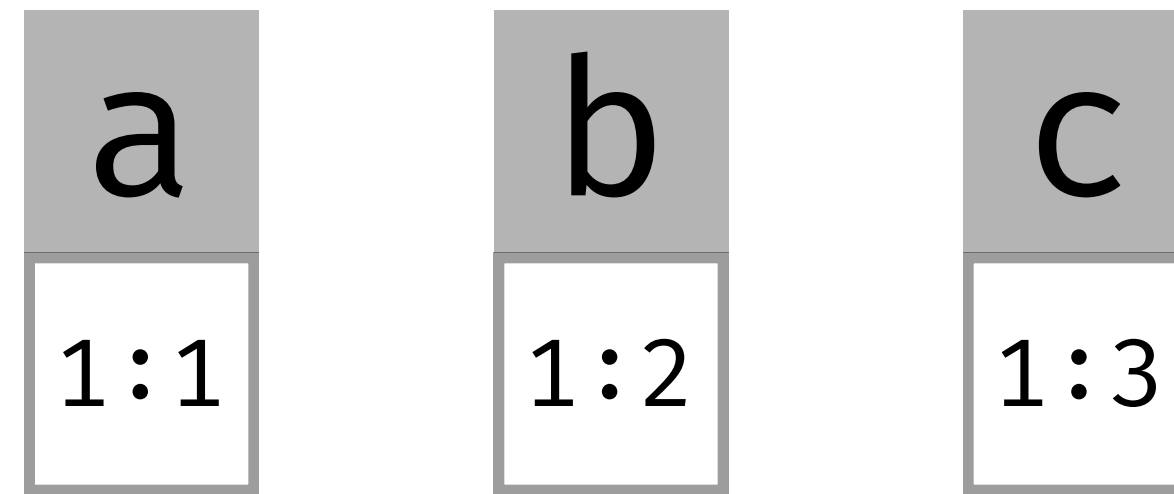
WOOT

2006

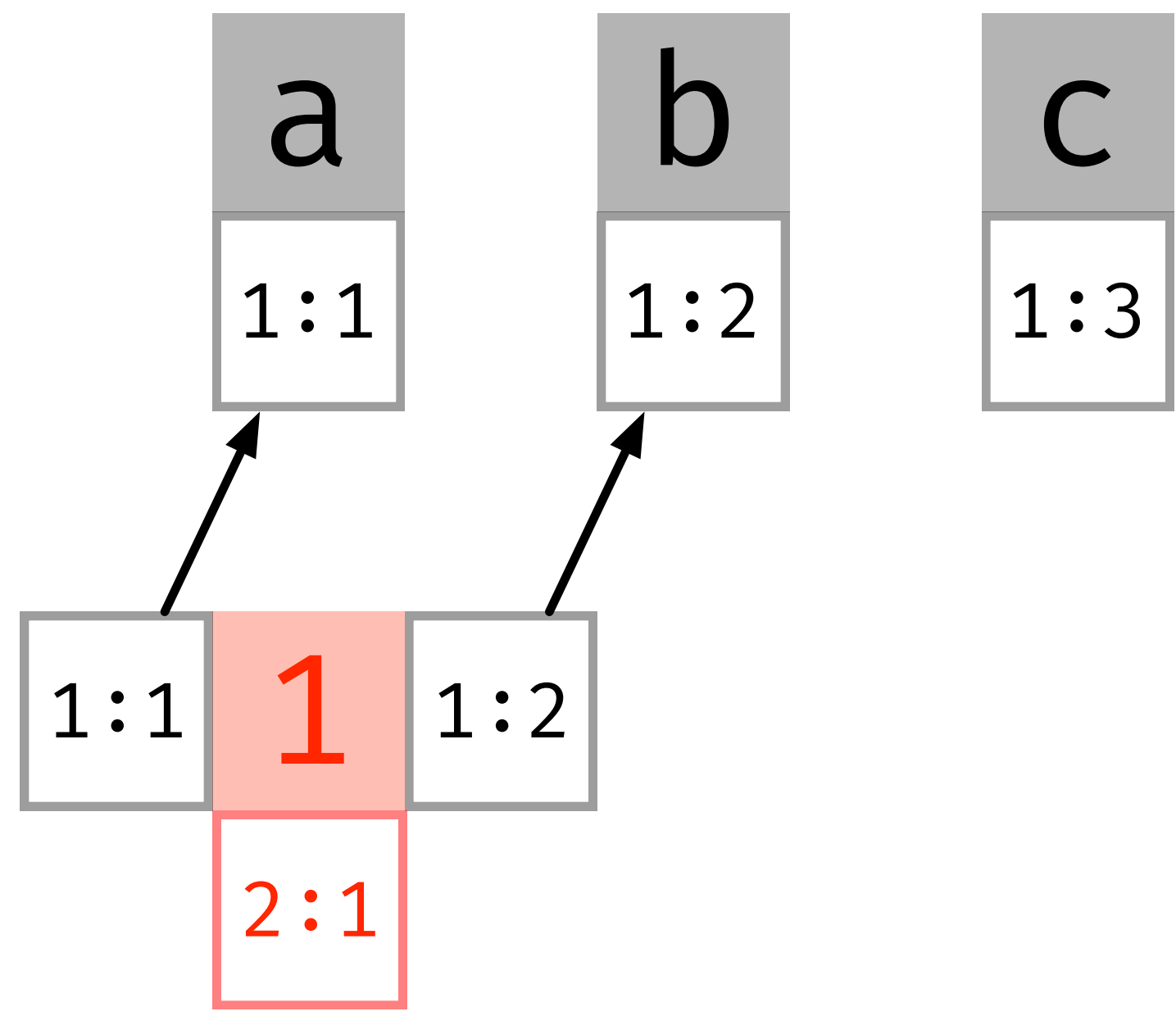
Without Operational Transformation

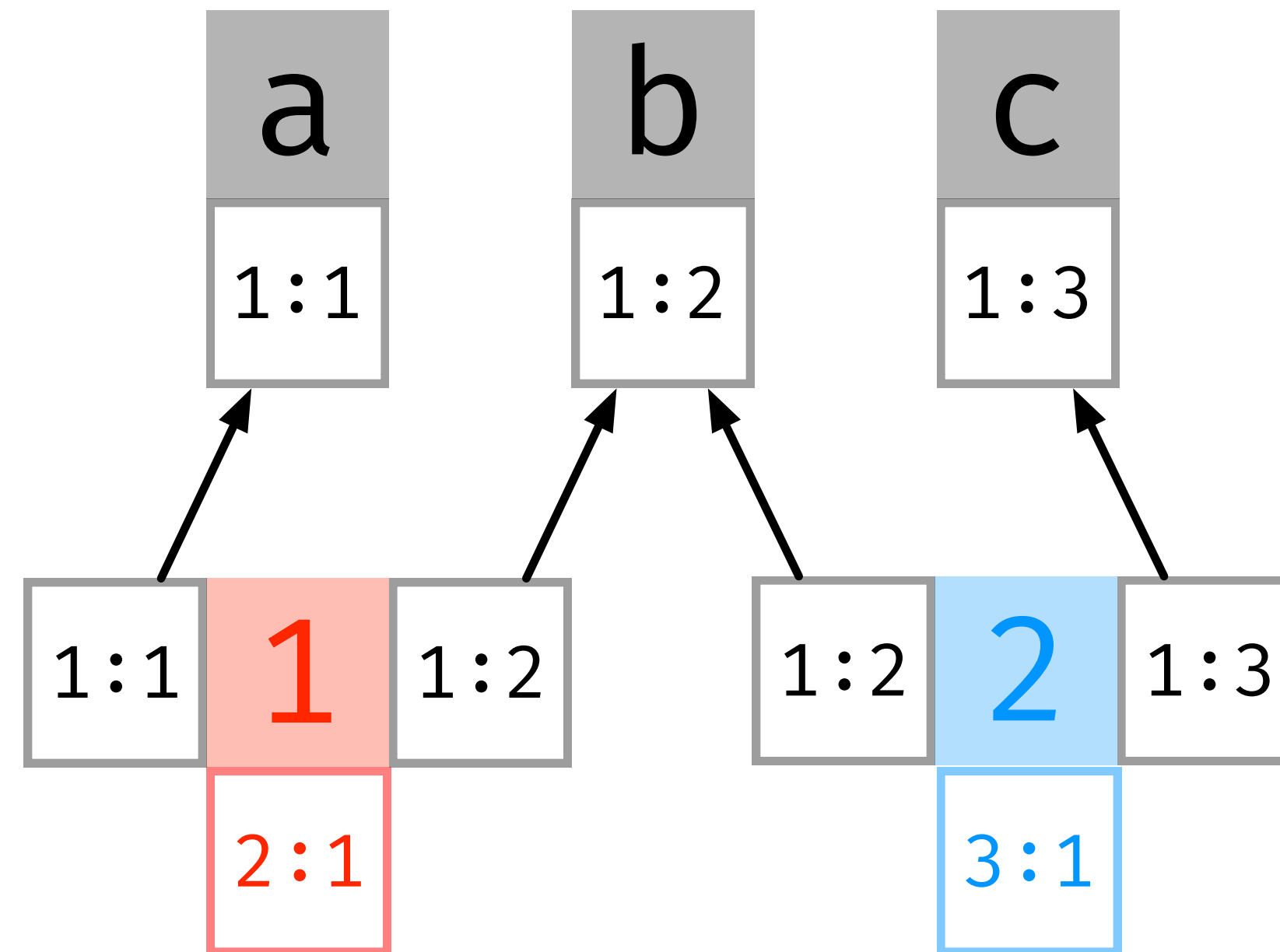
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

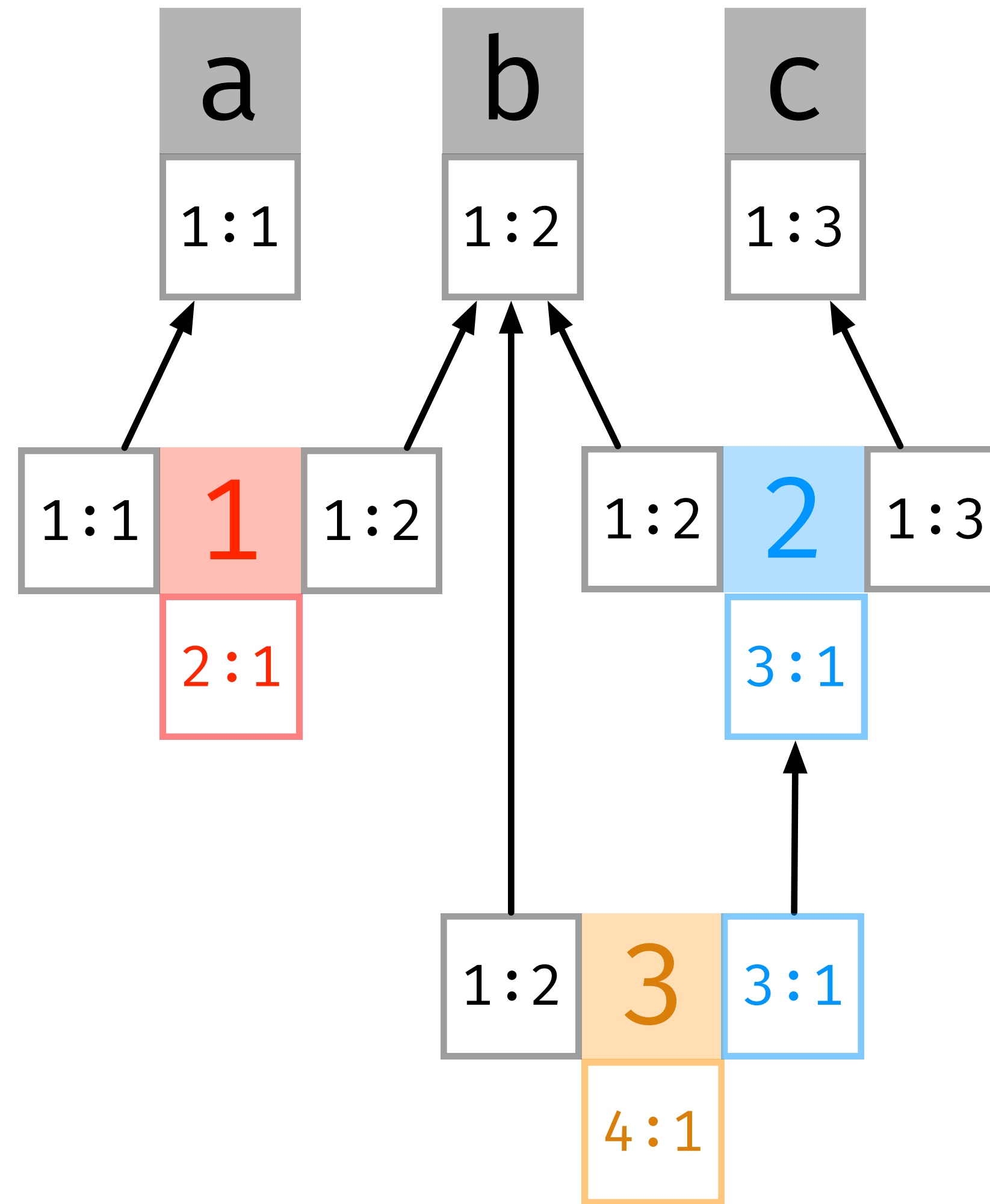
CSCW'06, November 4–8, 2006, Banff, Alberta, Canada.
Copyright 2006 ACM 1-59593-249-6/06/0011 ...\$5.00.



globally unique id
site-id:sequence-number







a
1:1

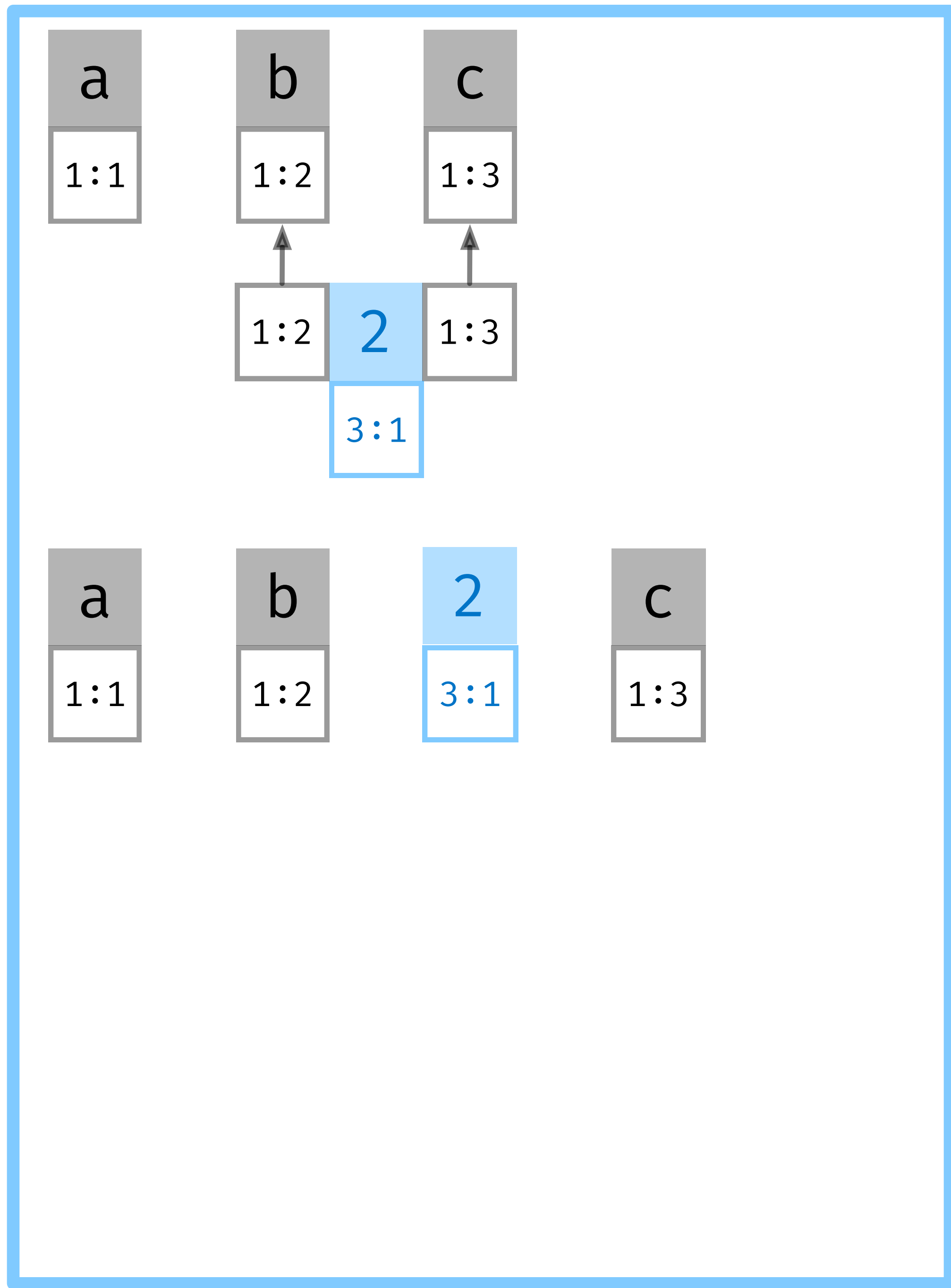
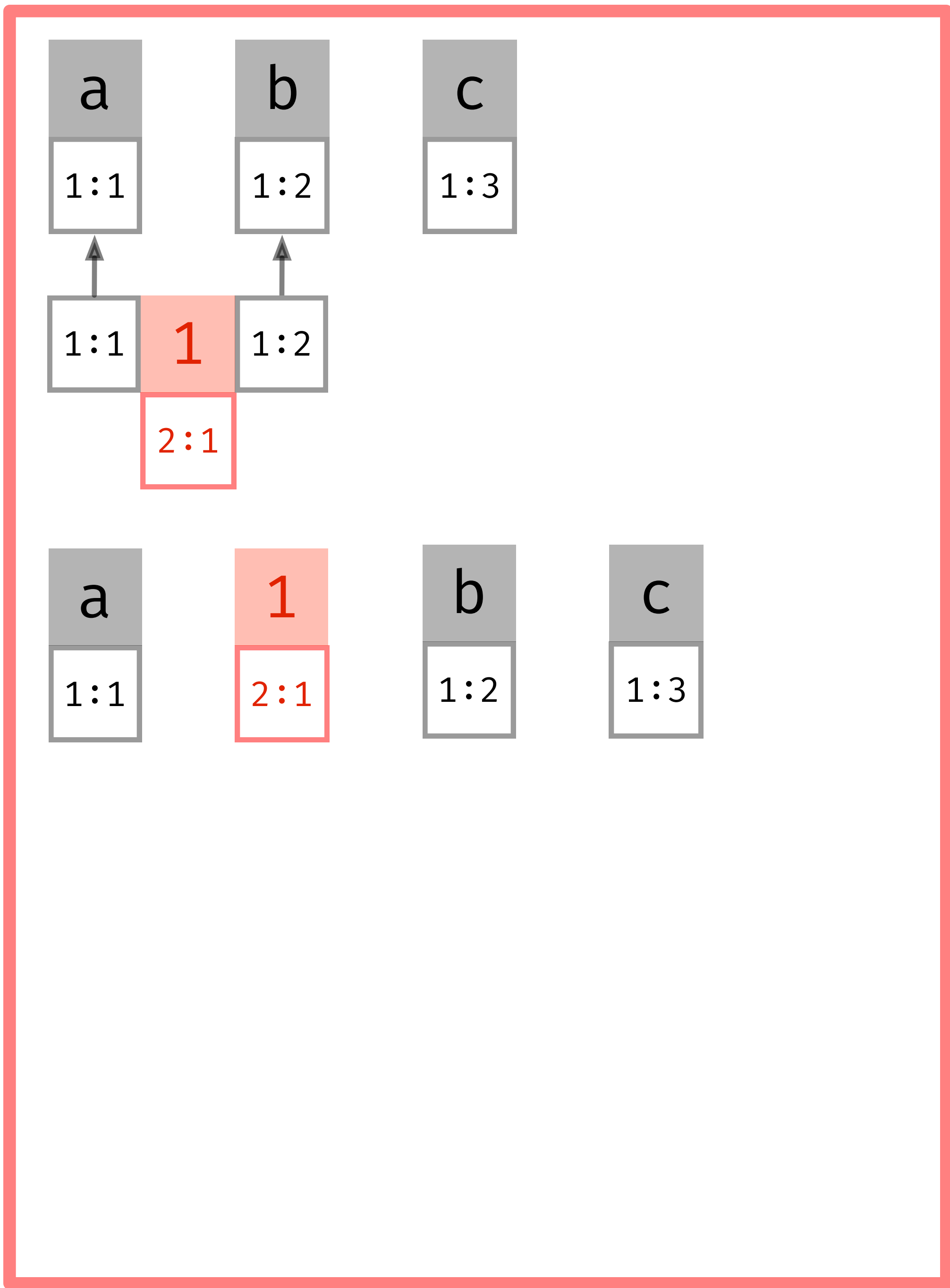
b
1:2

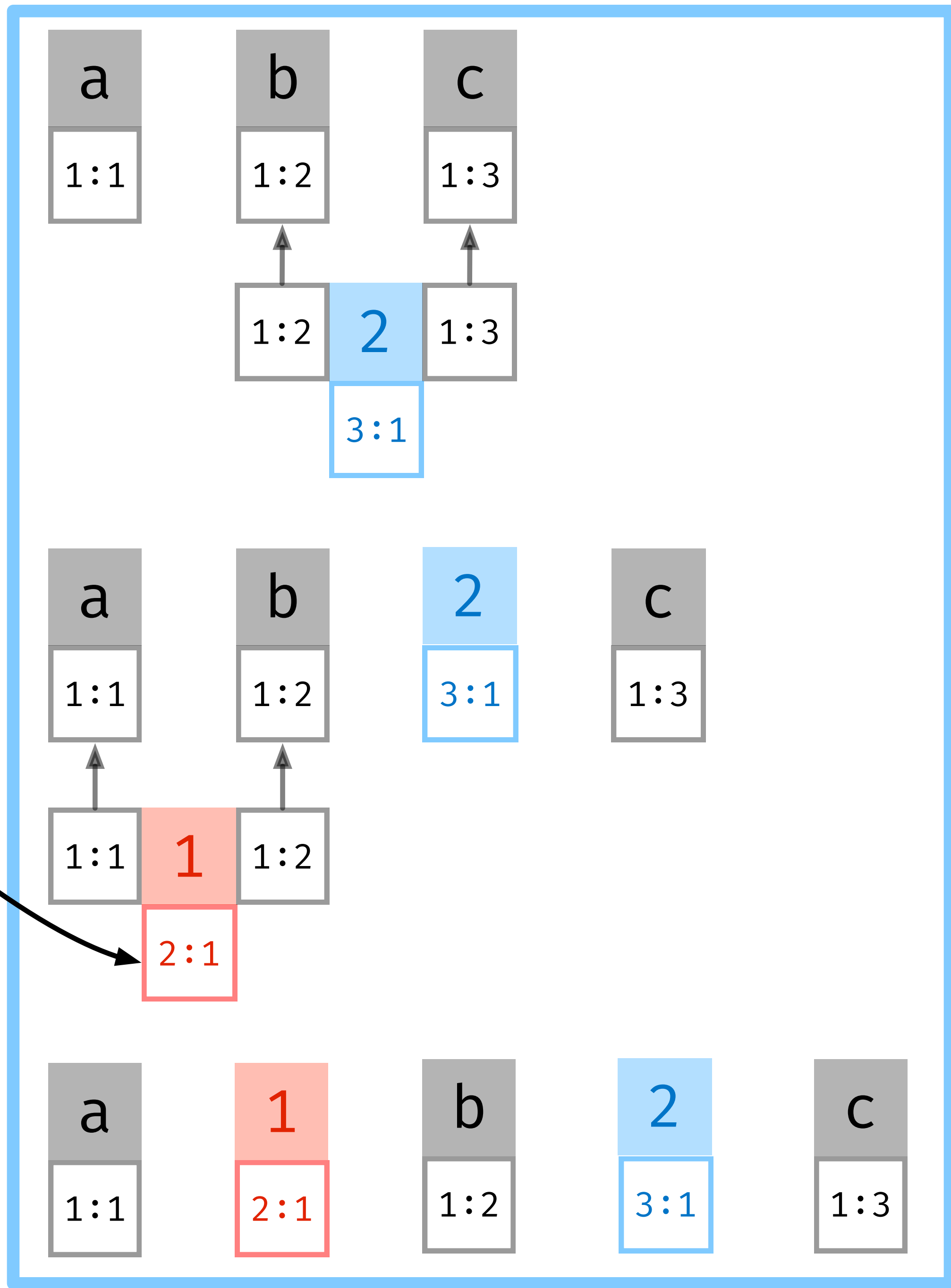
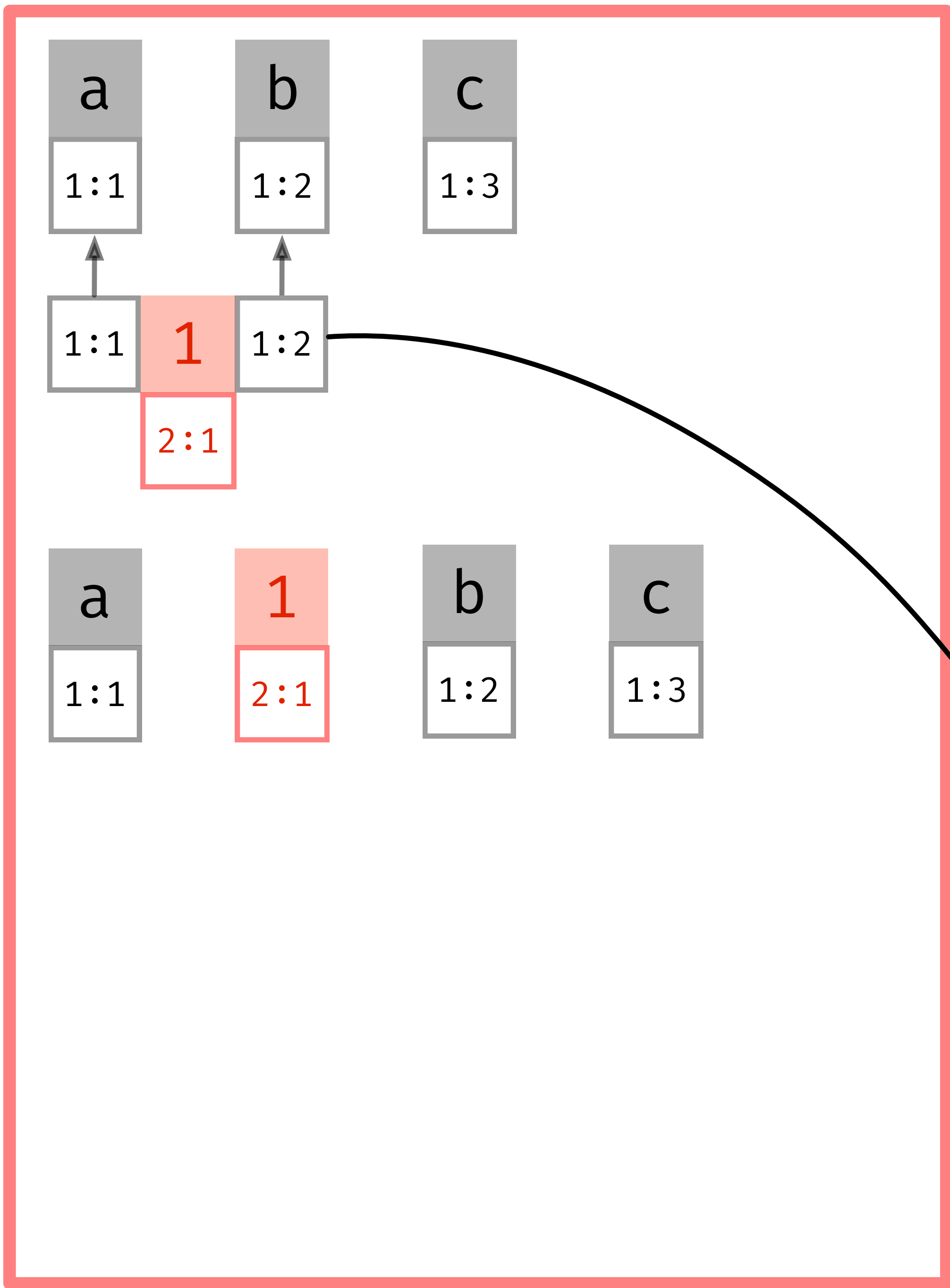
c
1:3

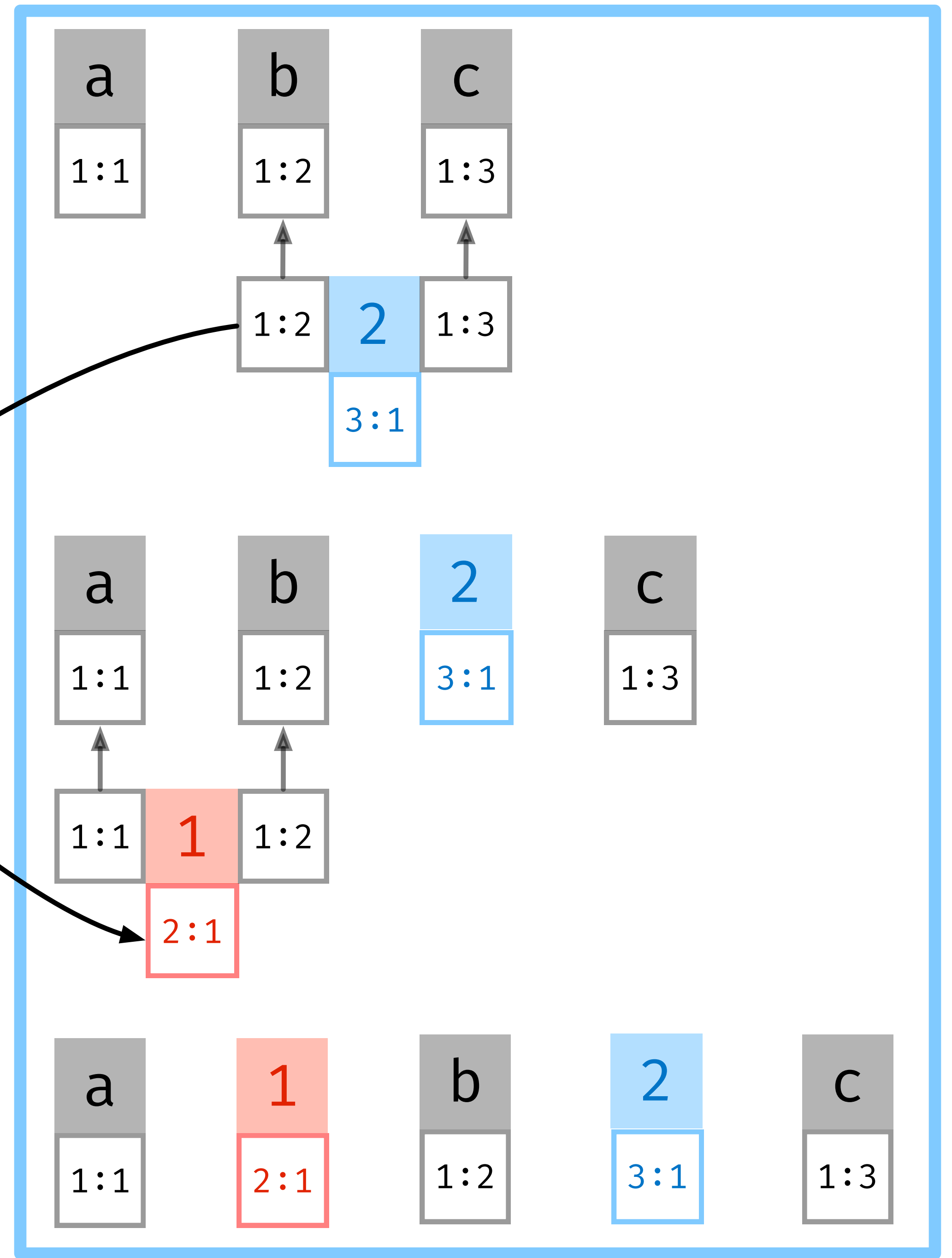
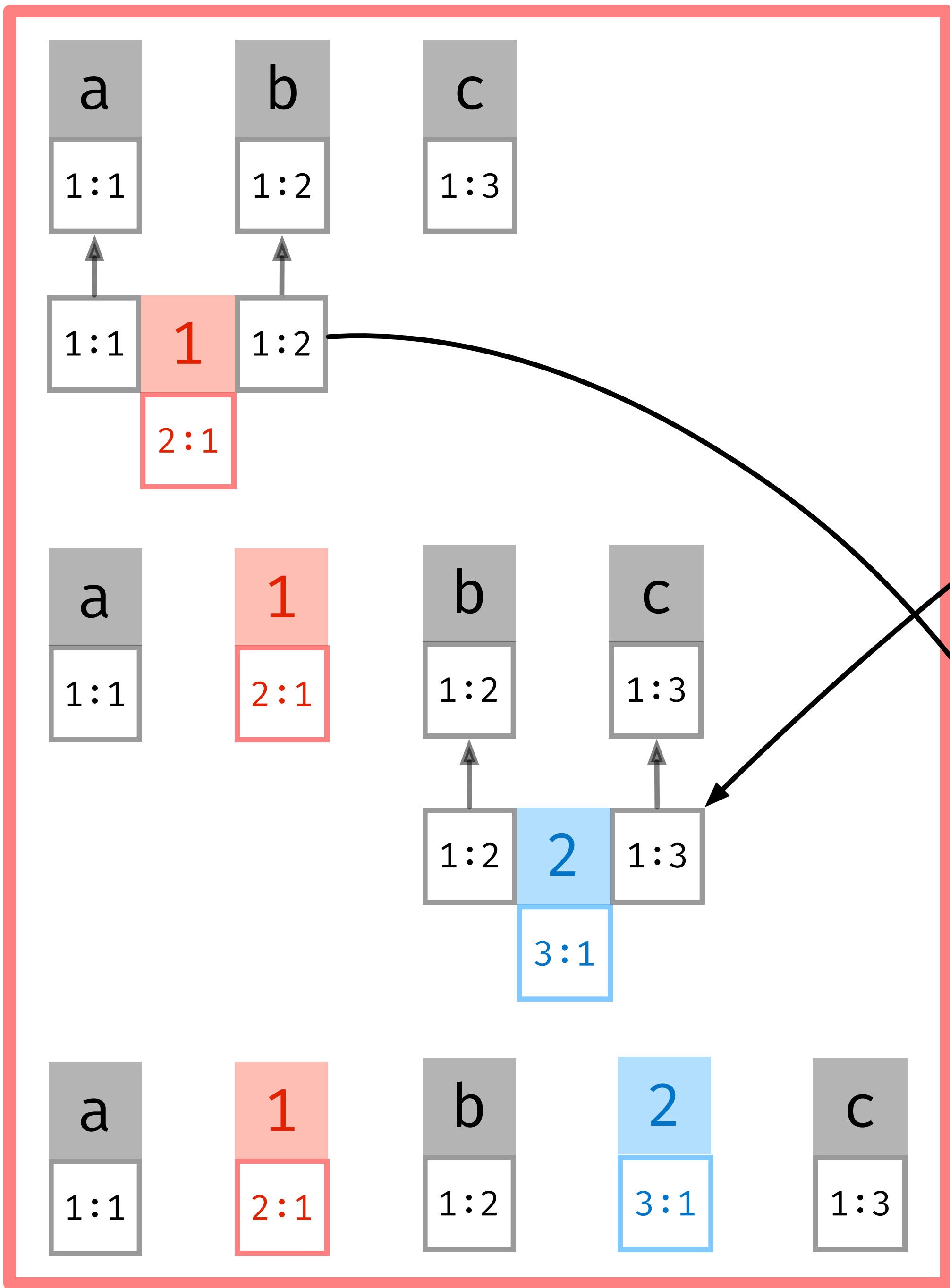
a
1:1

b
1:2

c
1:3

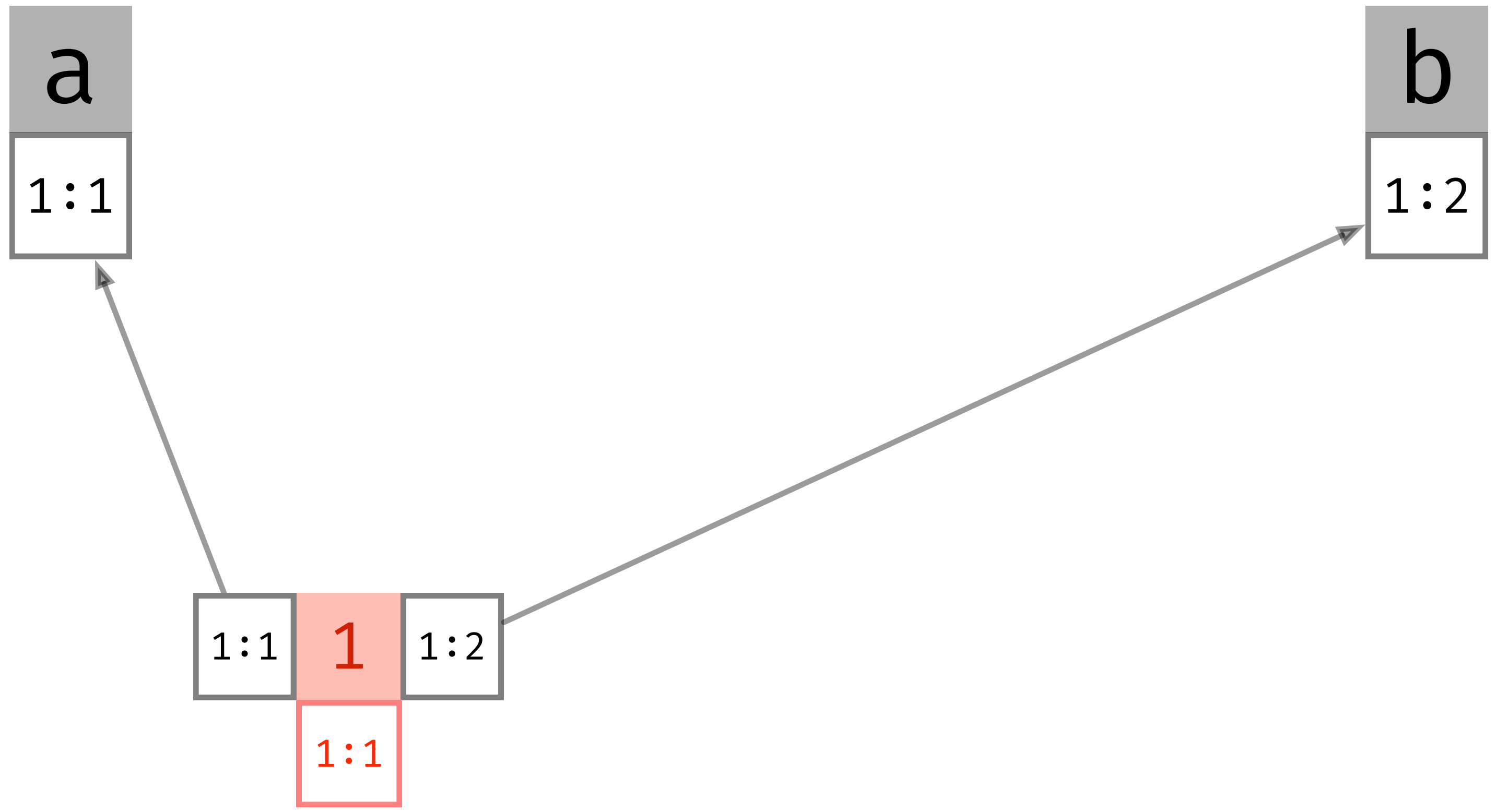


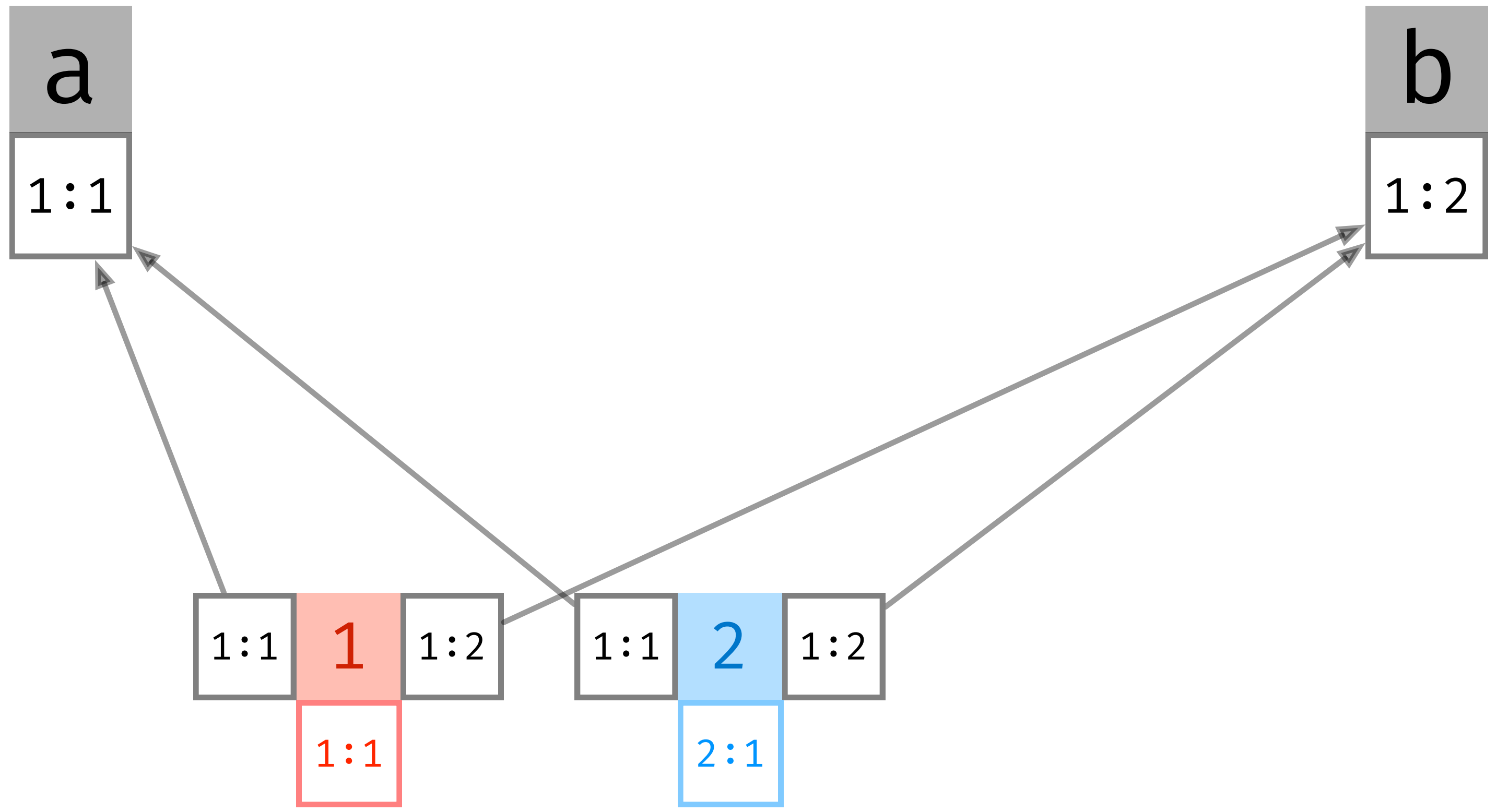


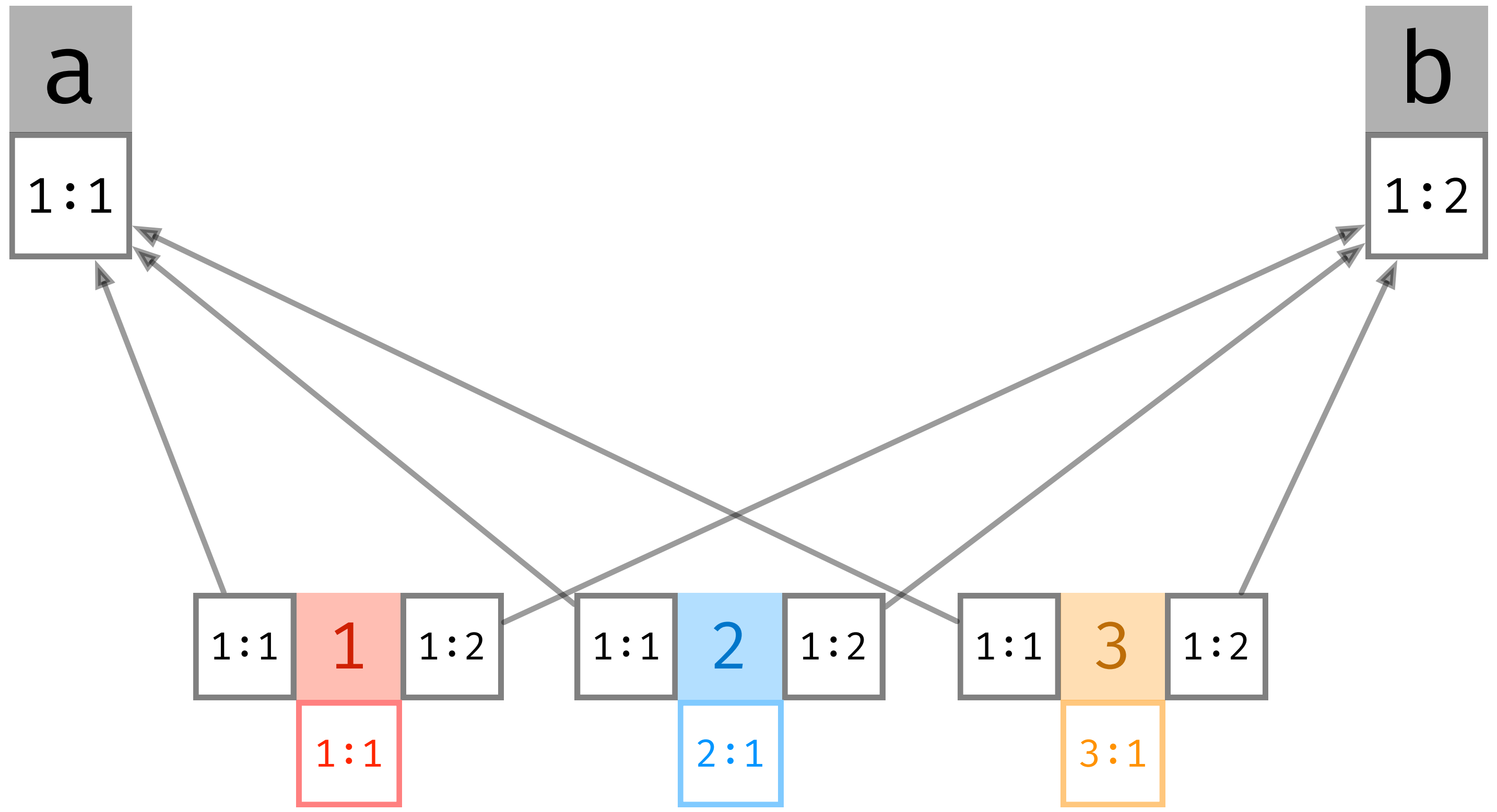


a
1:1

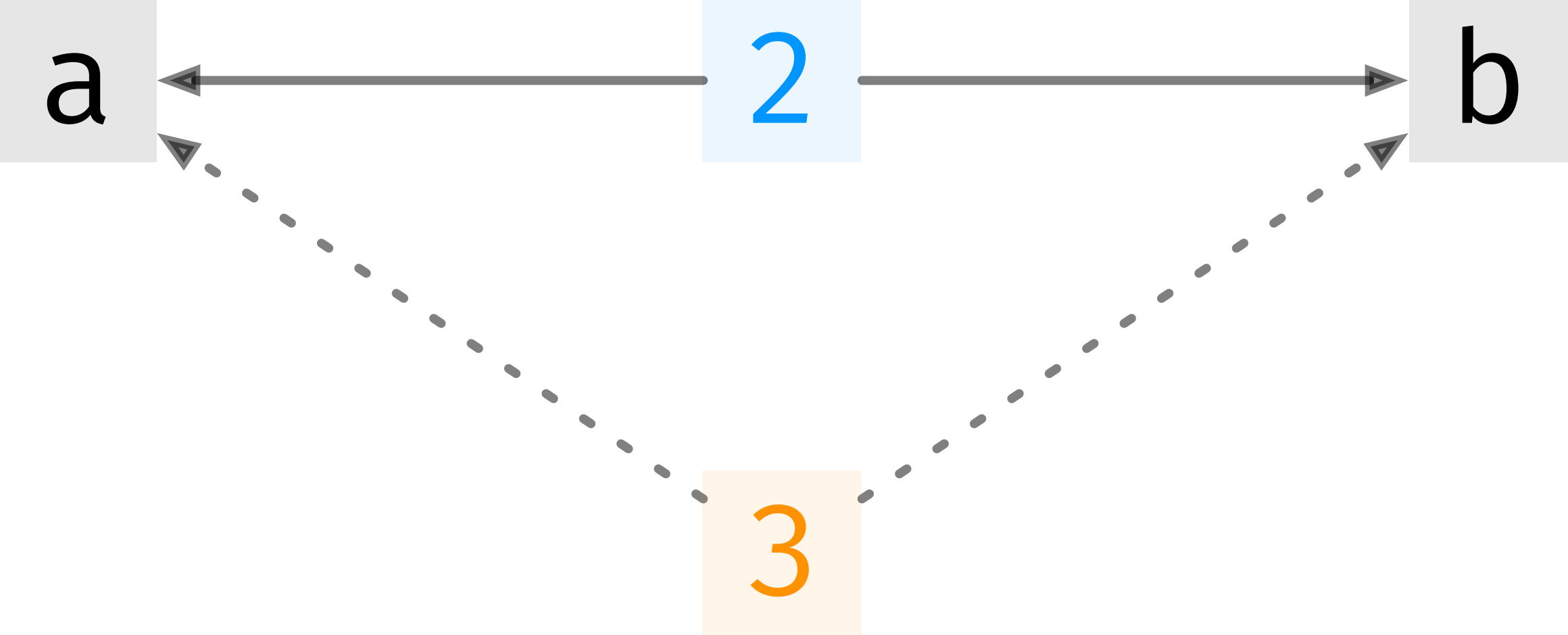
b
1:2

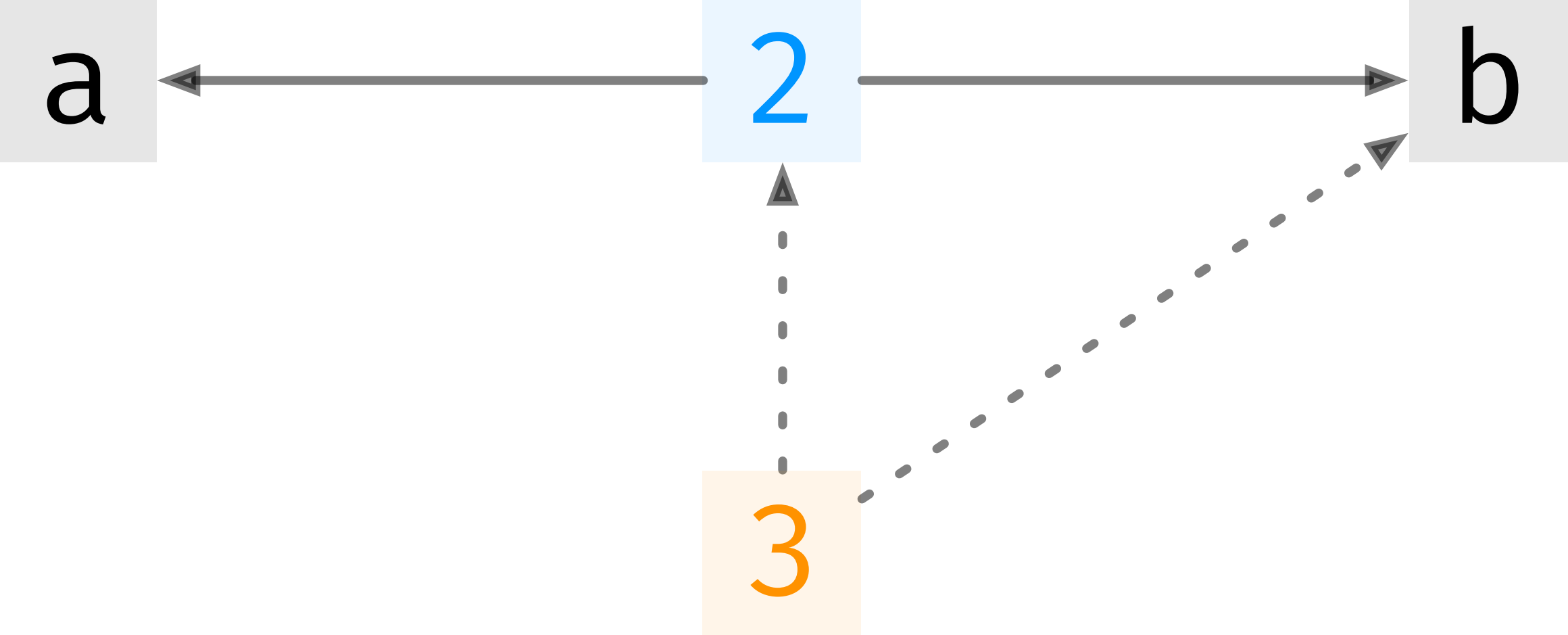


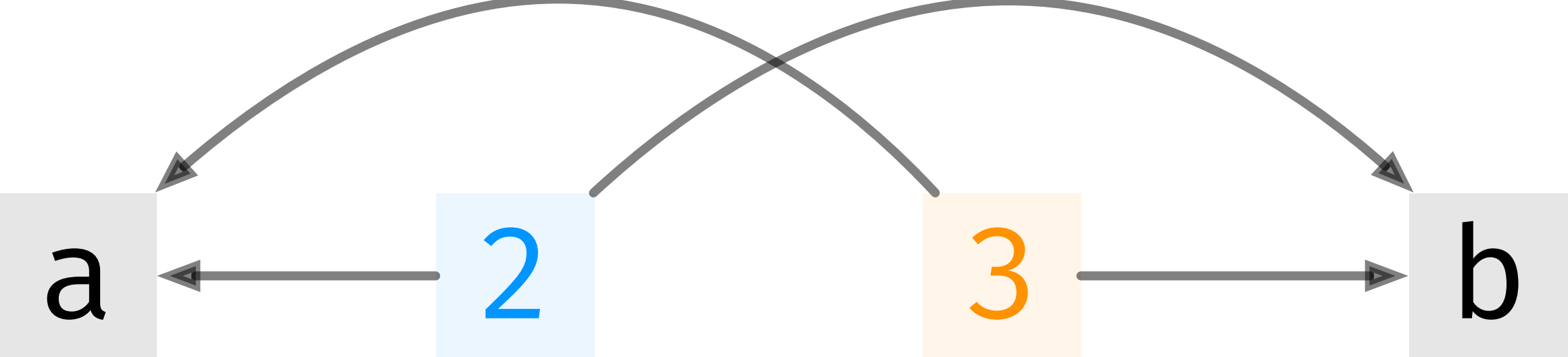


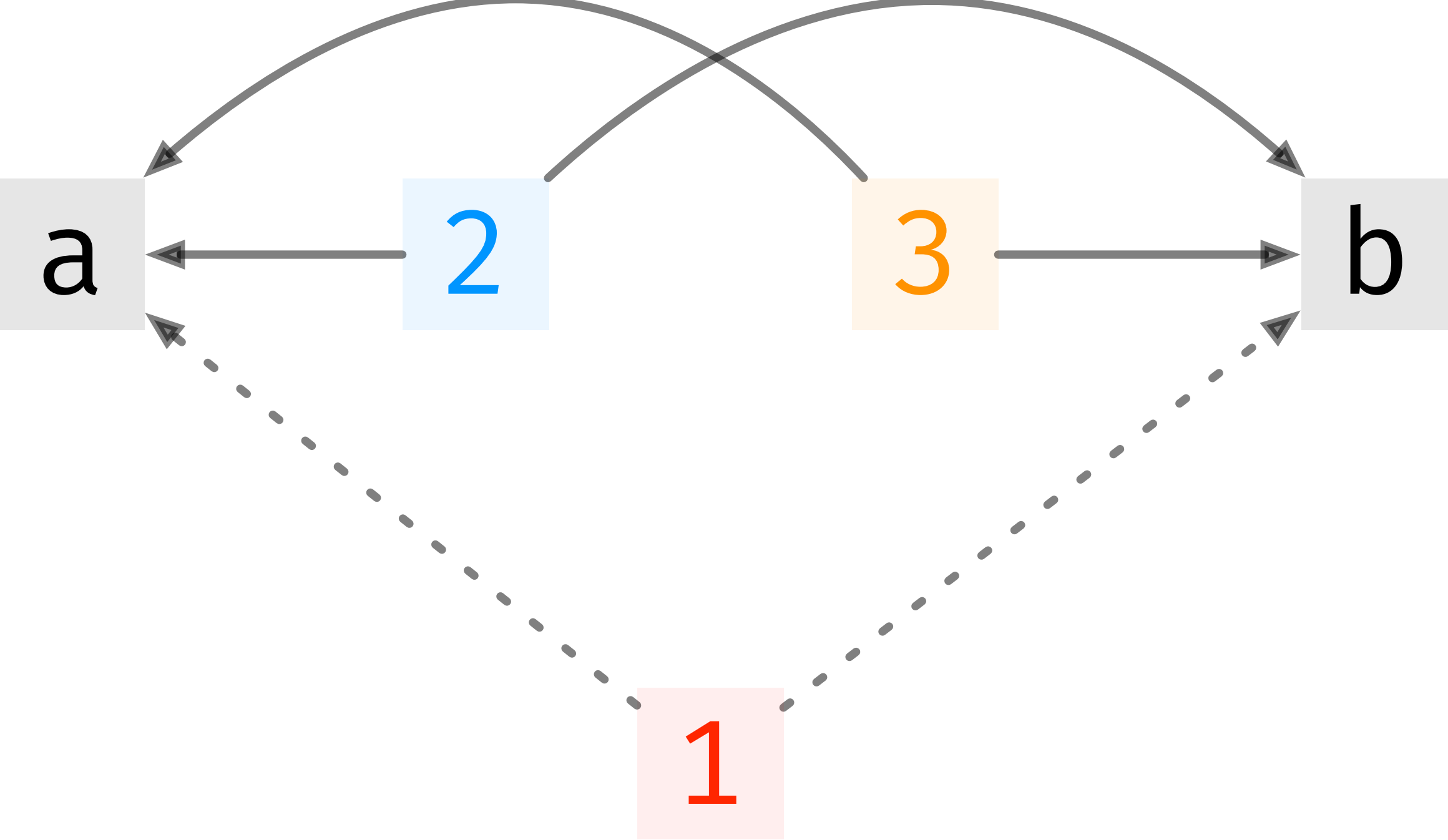


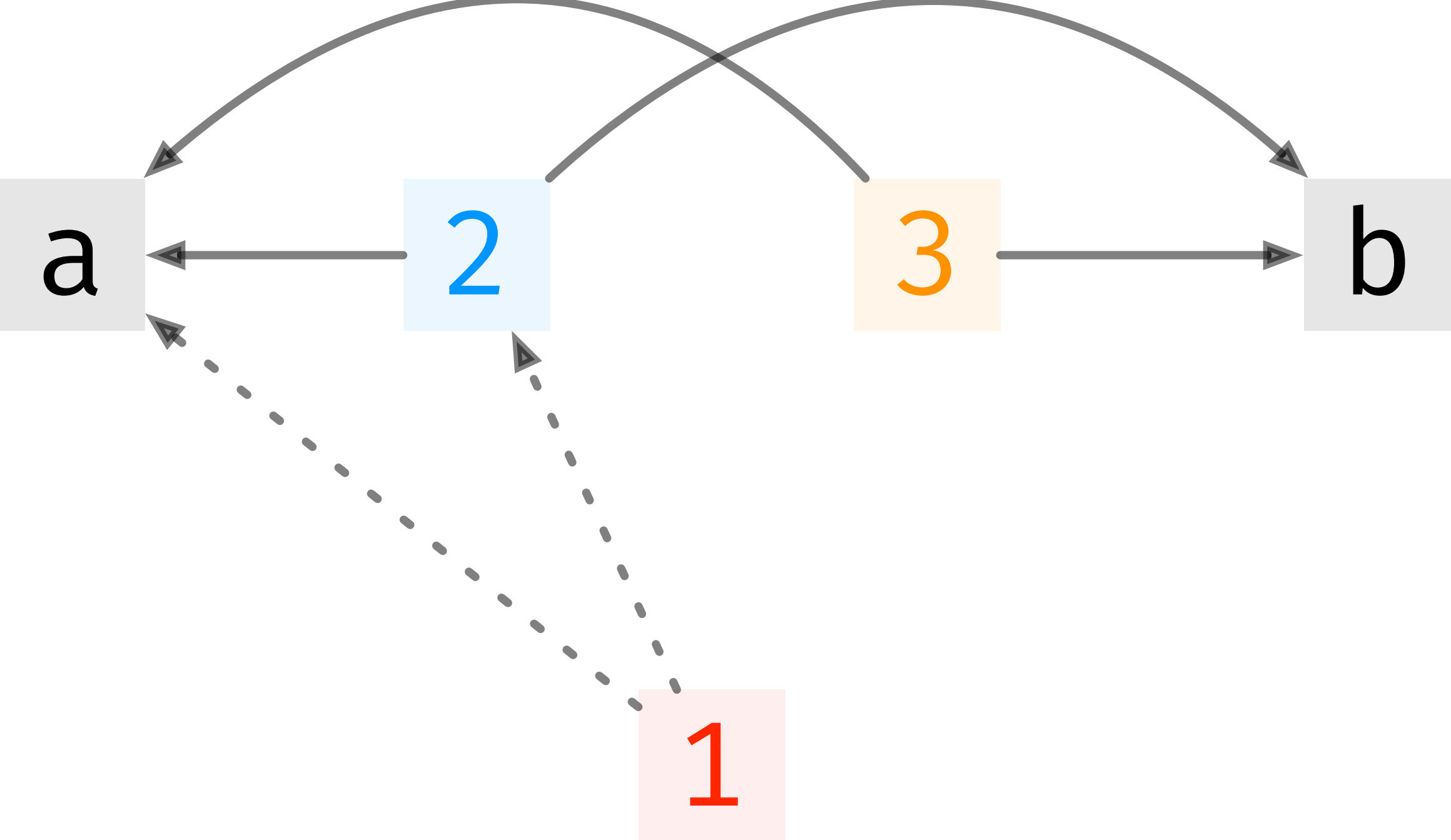


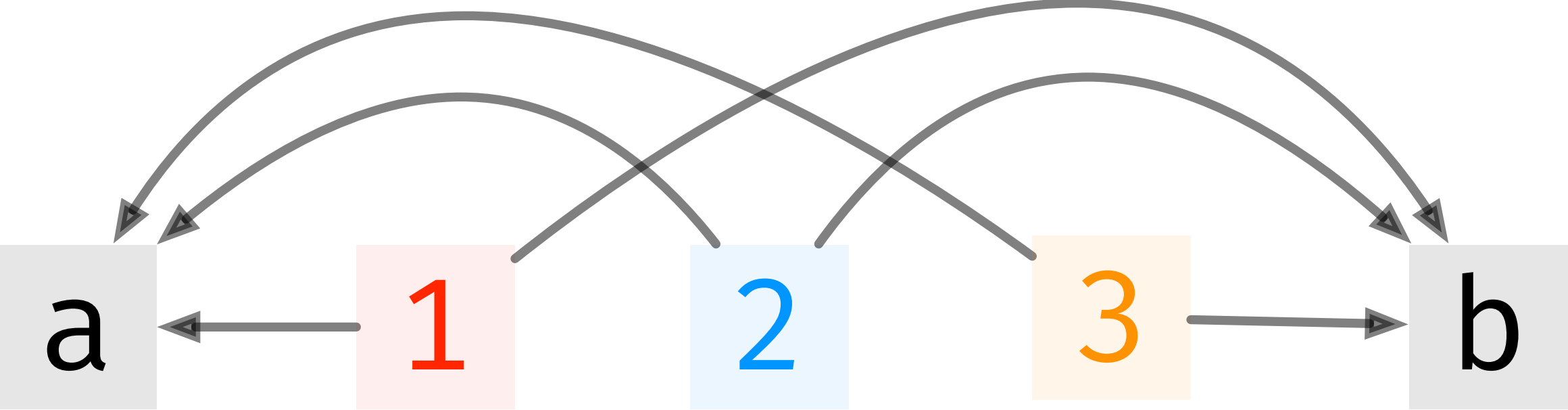














a b



a b



a b

a b

a < 1 < b

a 1 b

a b

a b

a < 2 < b

a 2 b

a b

a < 1 < b

a 1 b

a b

a < 1 < b

a 1 b

a b

a < 2 < b

a 2 b

a b

a < 1 < b

a 1 b

a b

a < 1 < b

a 1 b

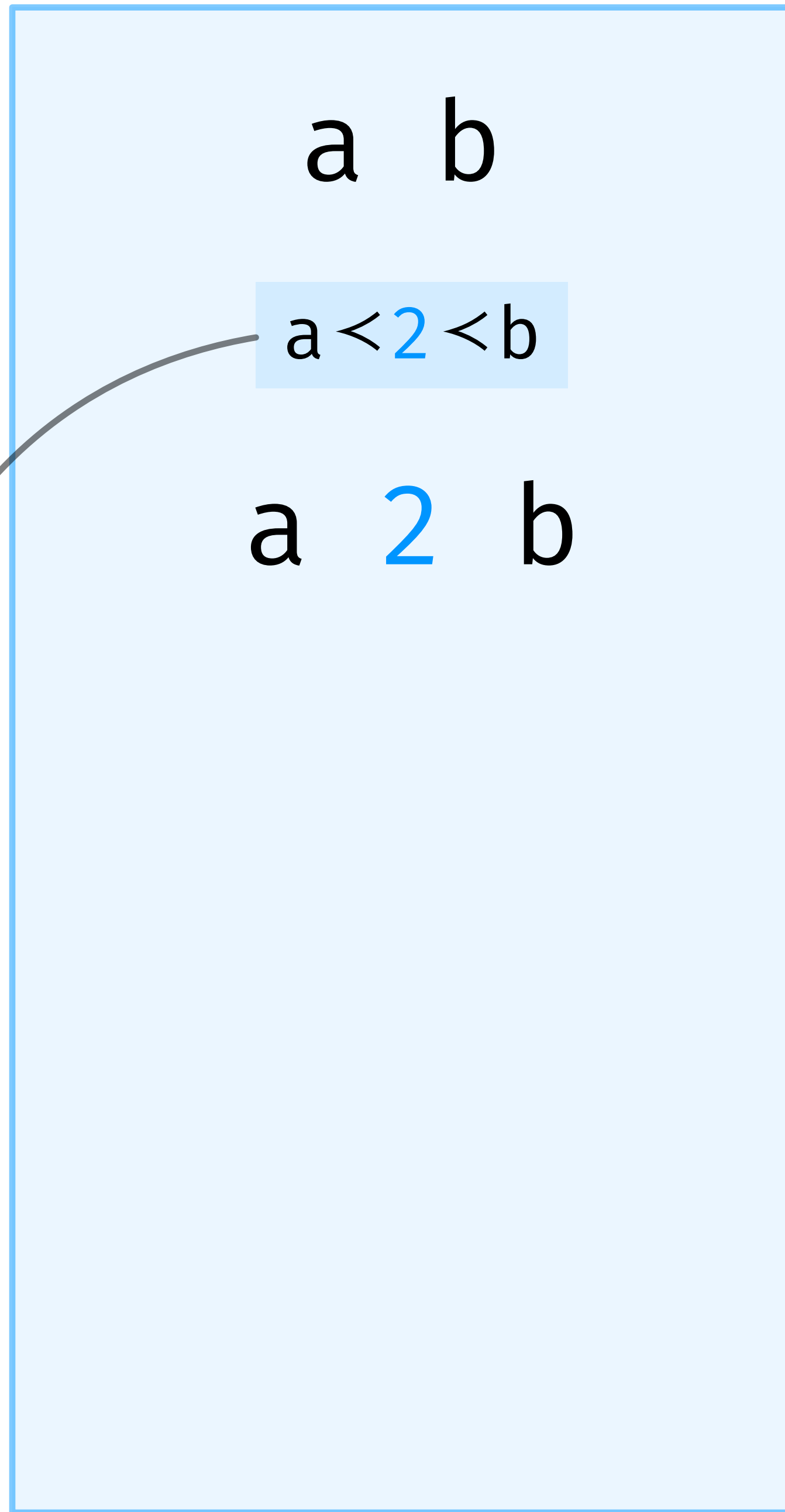
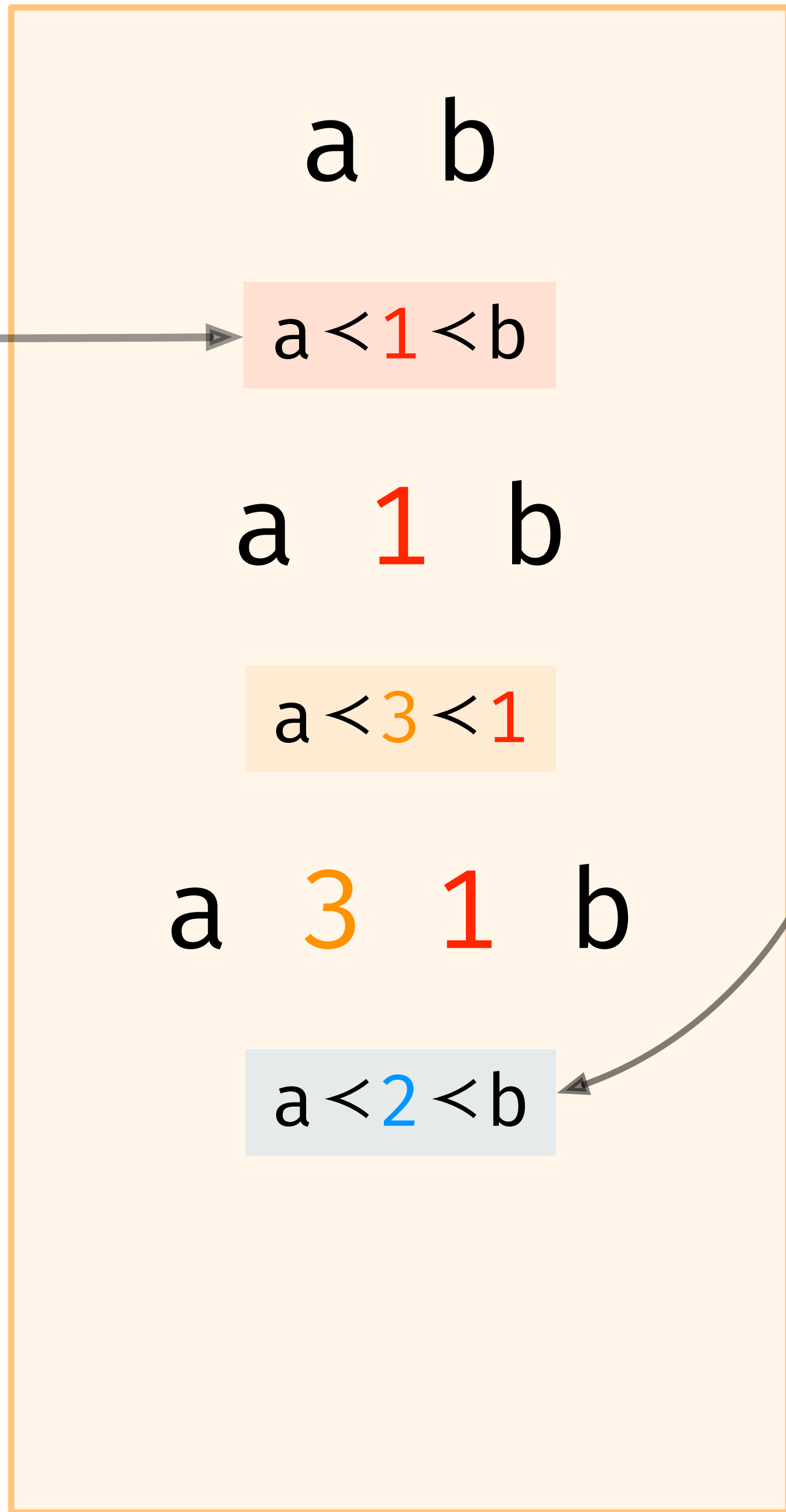
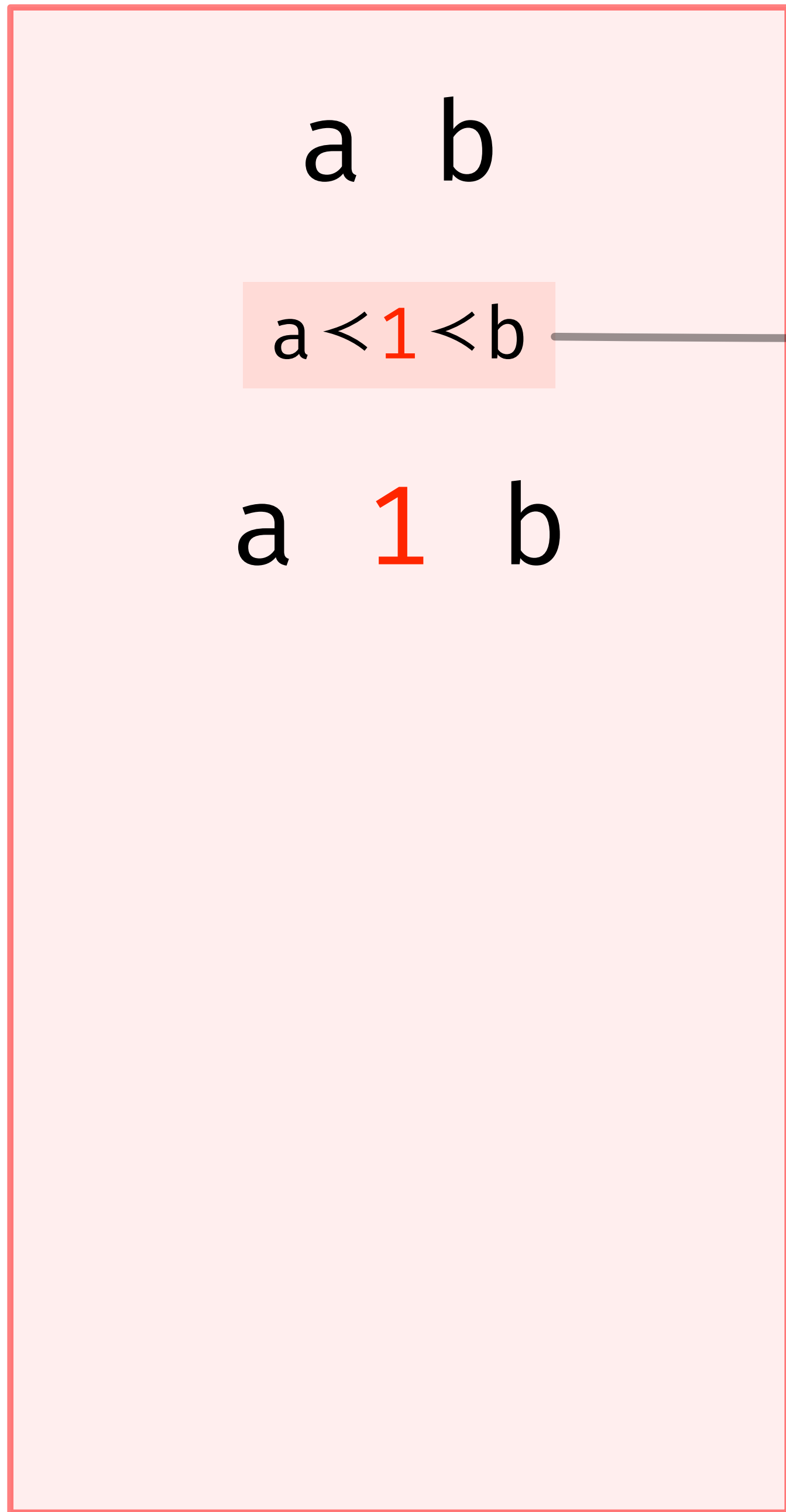
a < 3 < 1

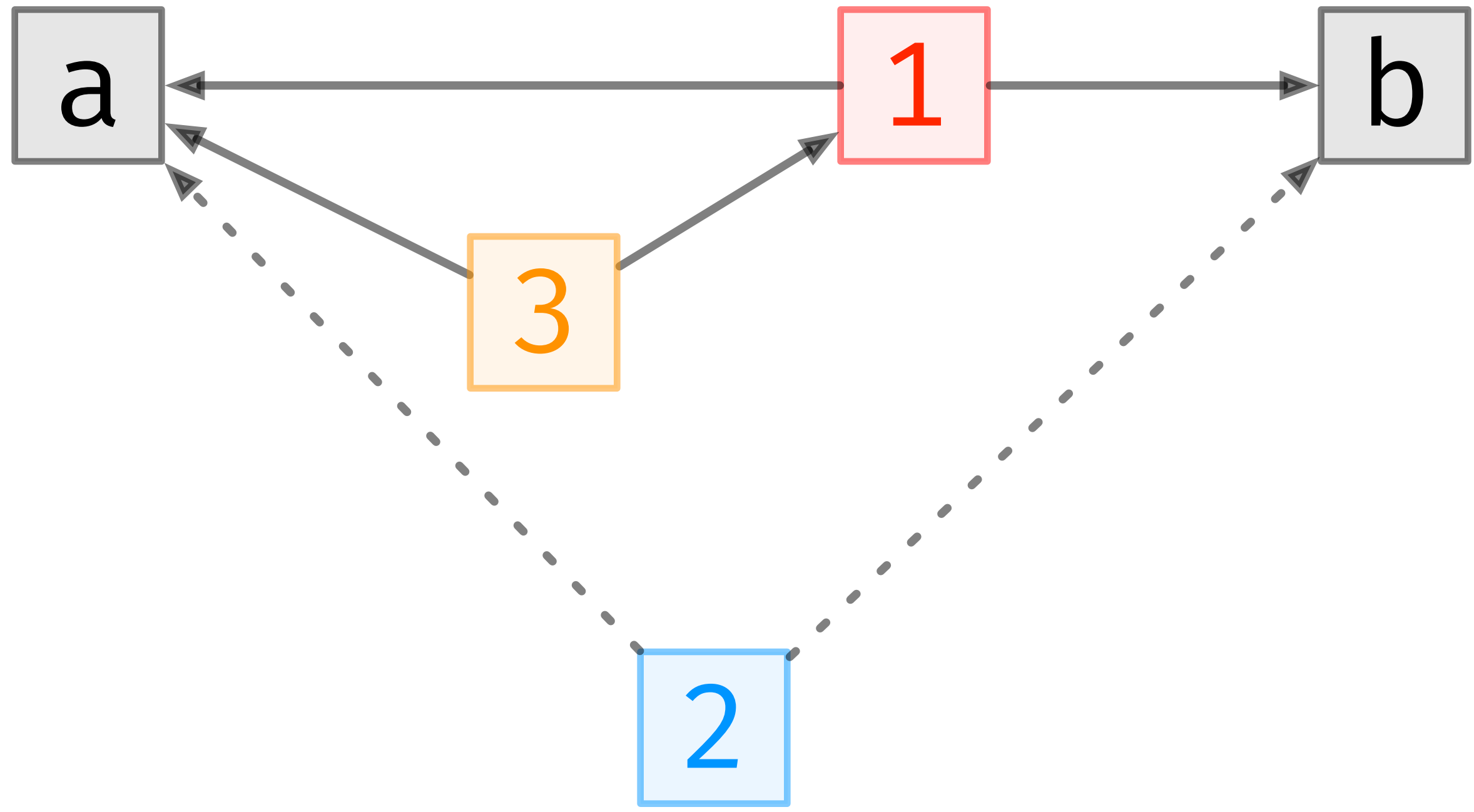
a 3 1 b

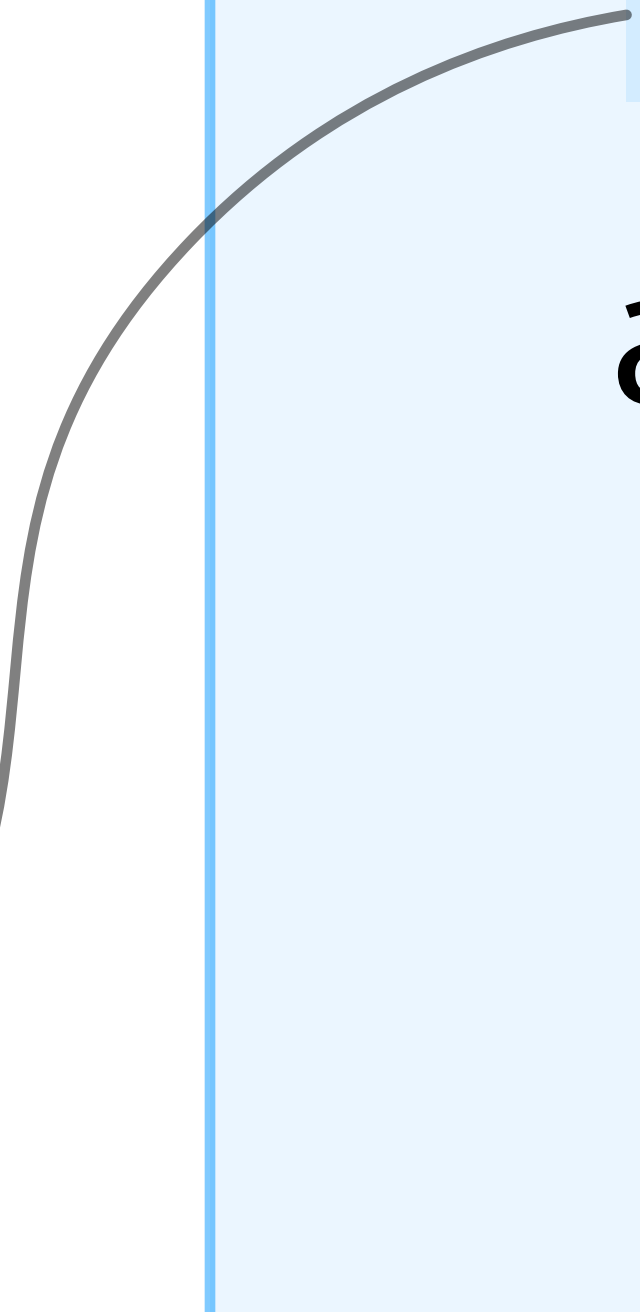
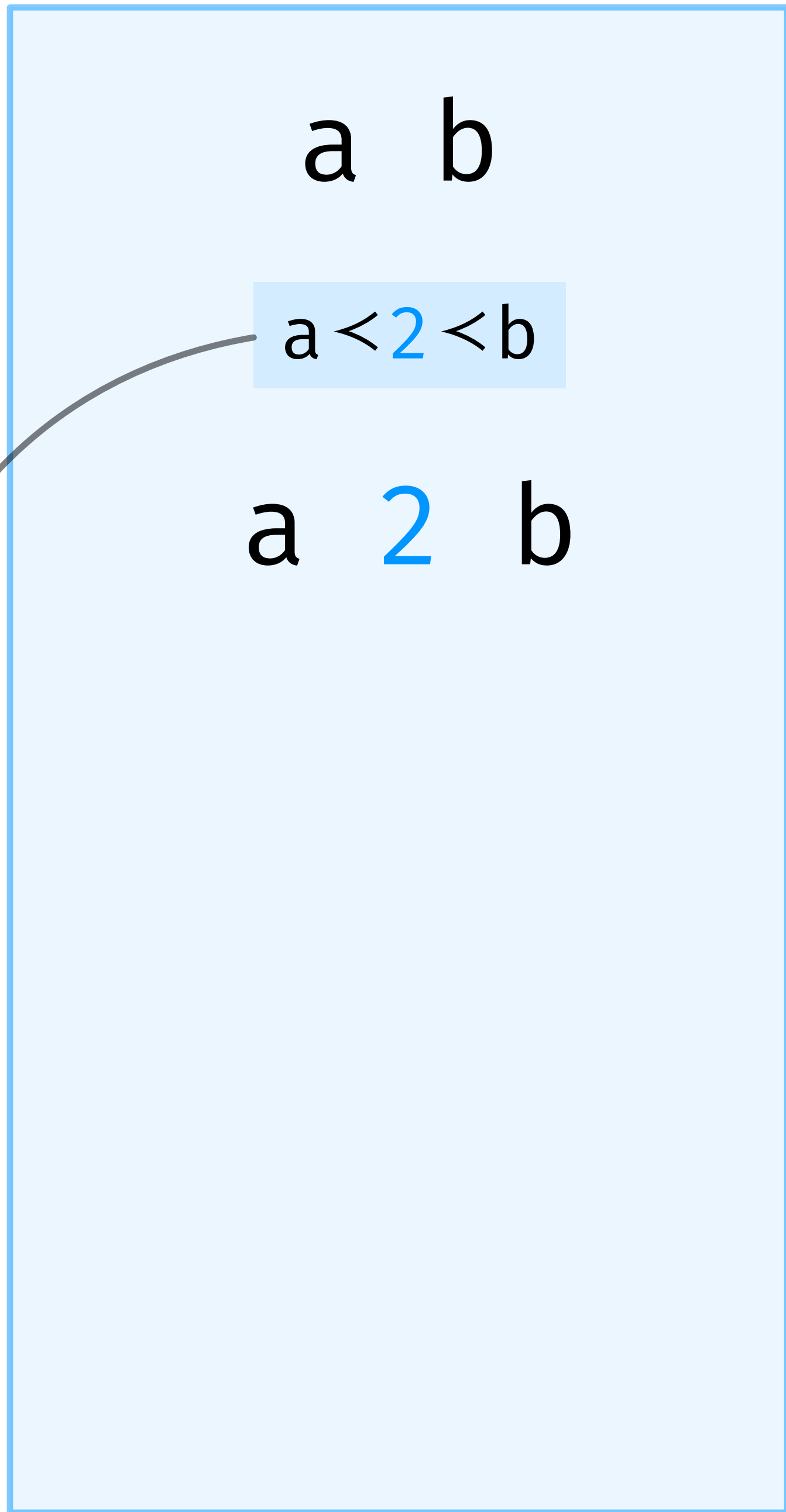
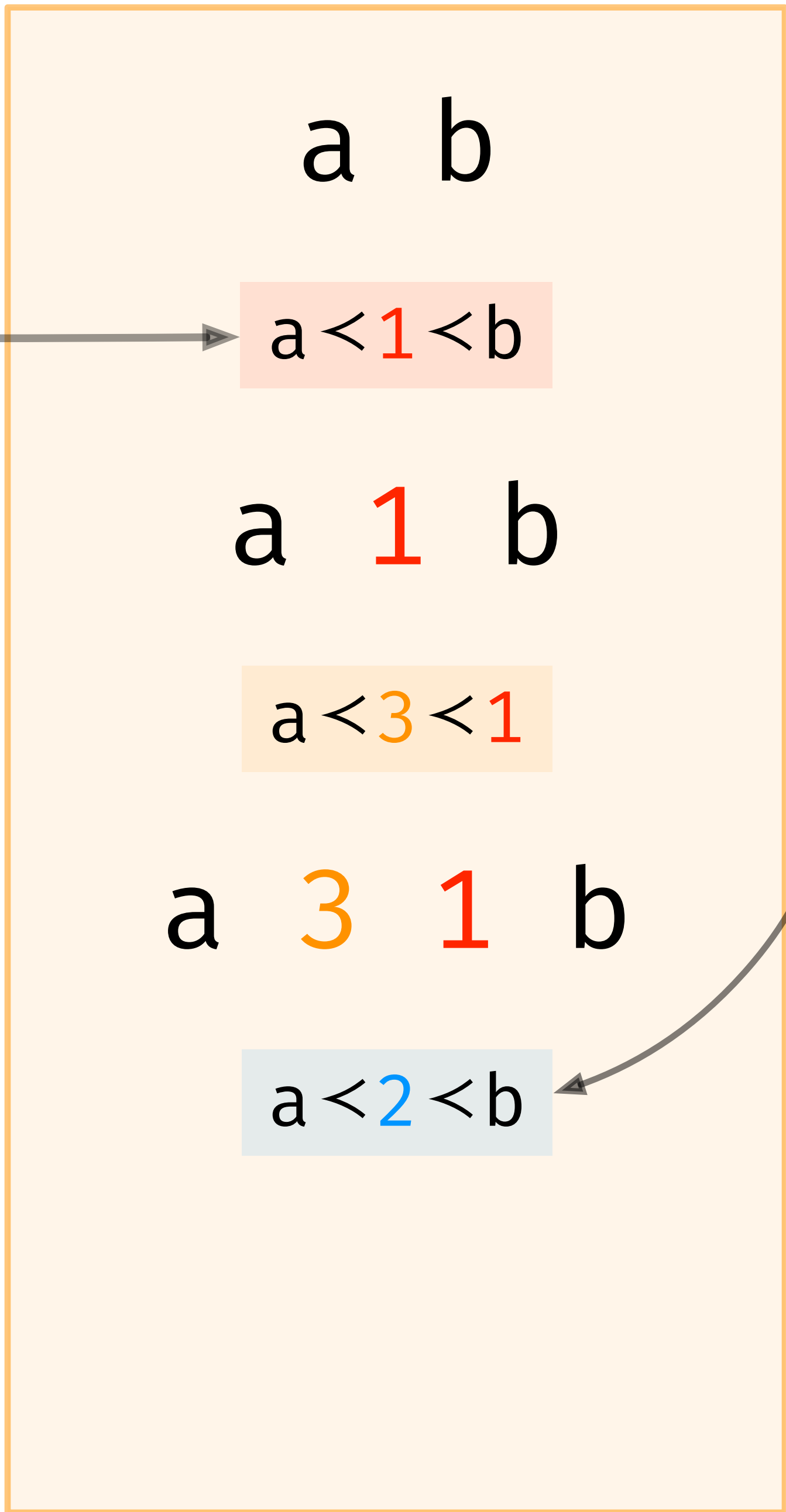
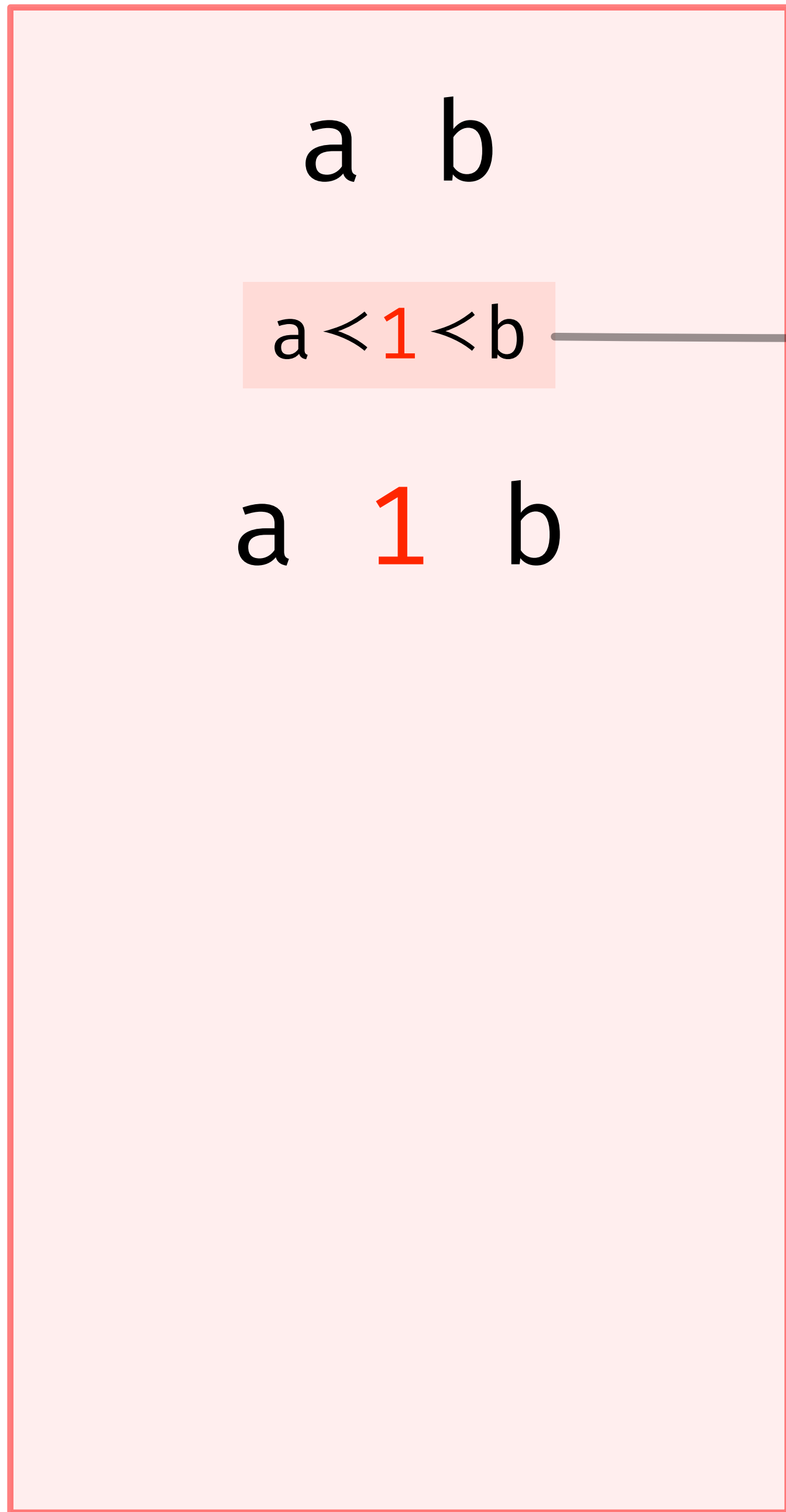
a b

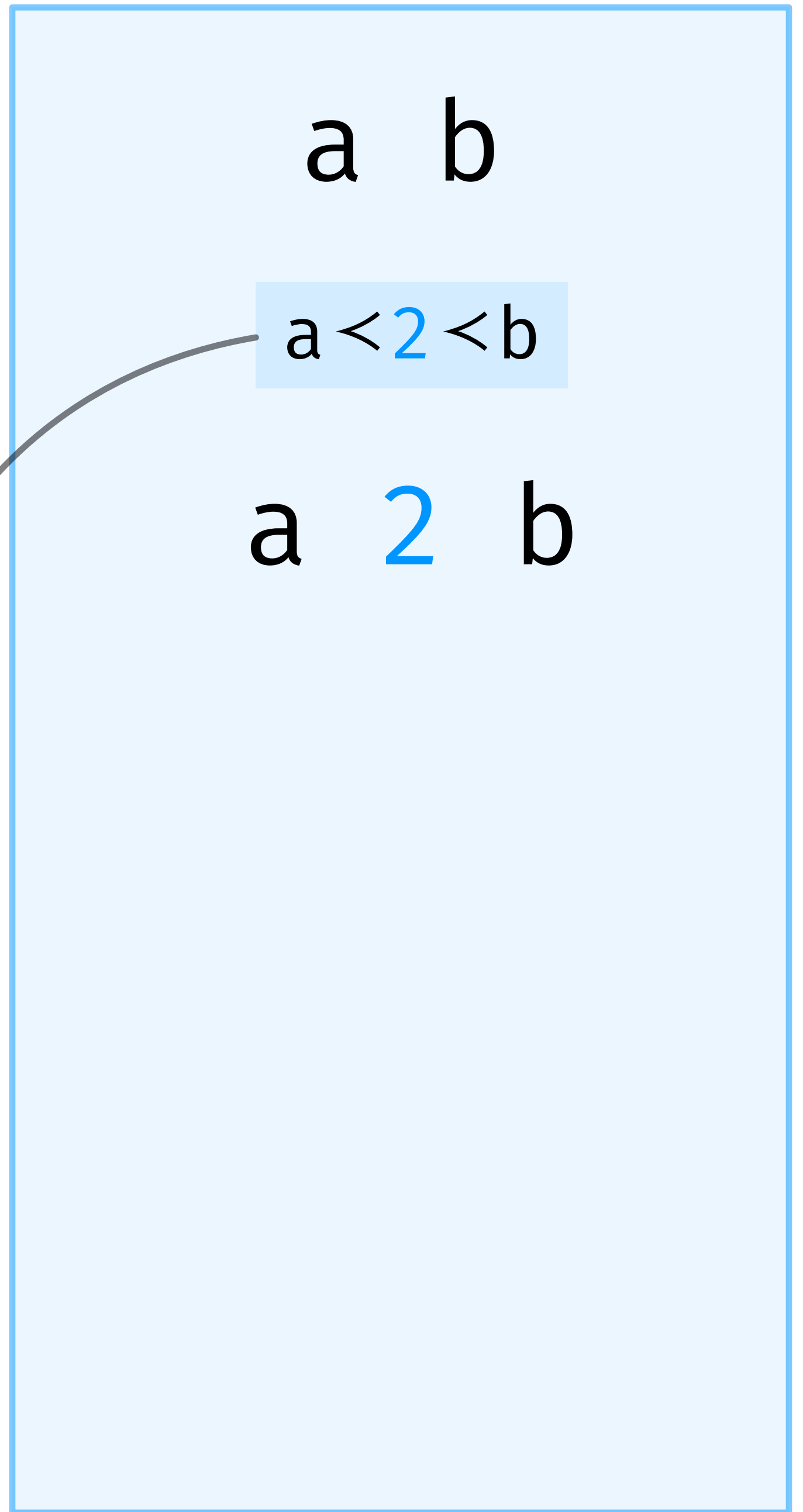
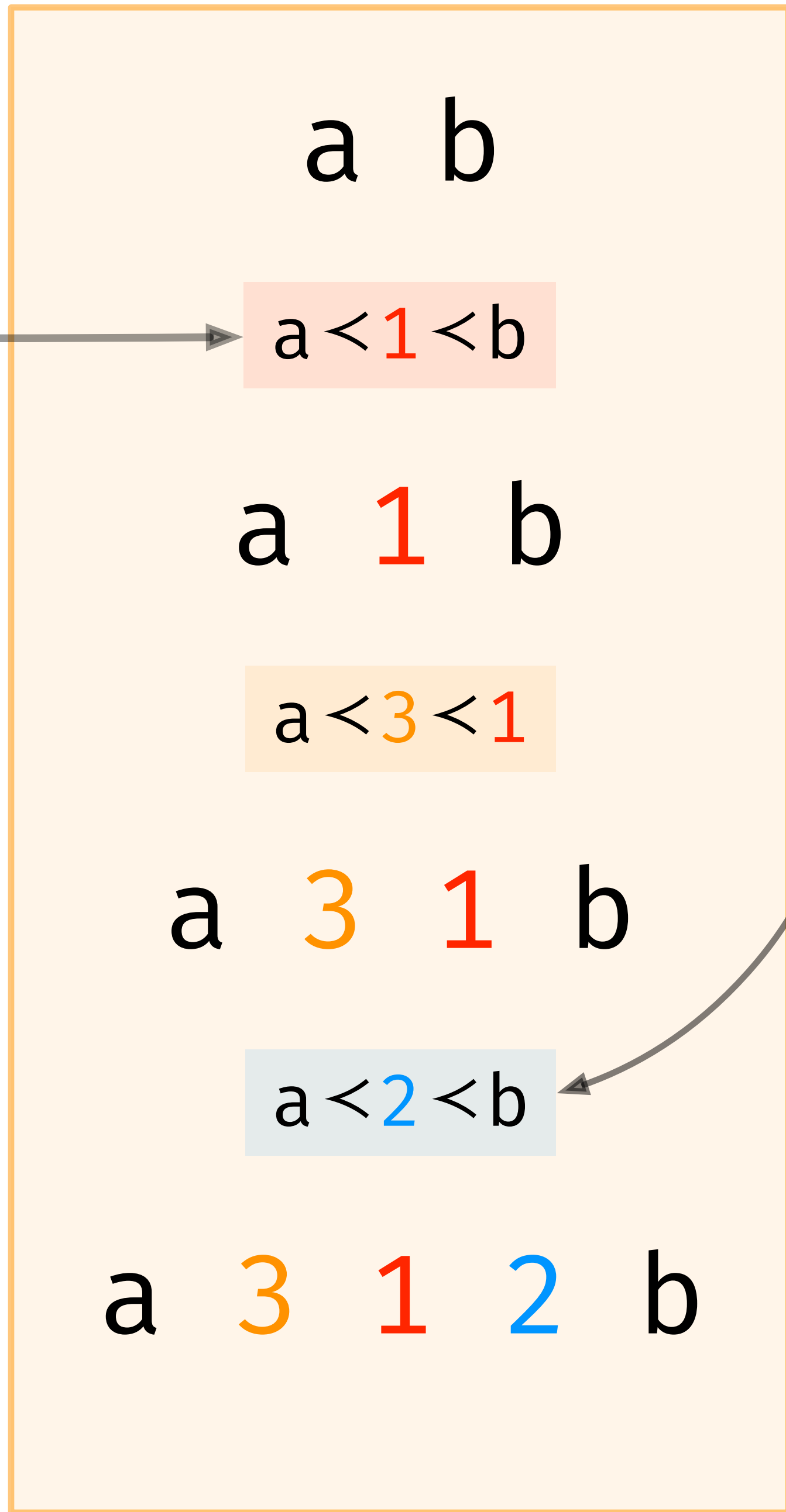
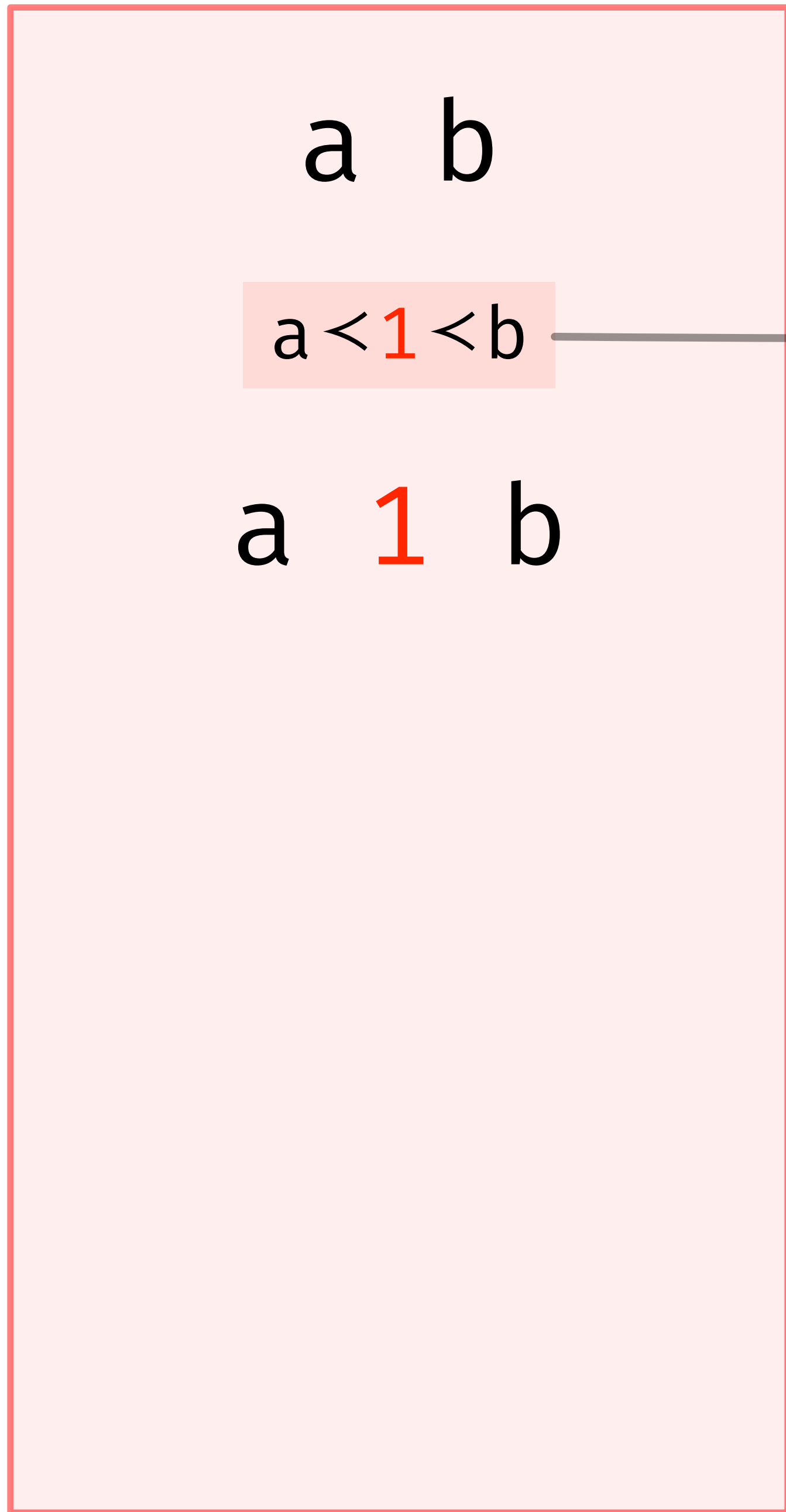
a < 2 < b

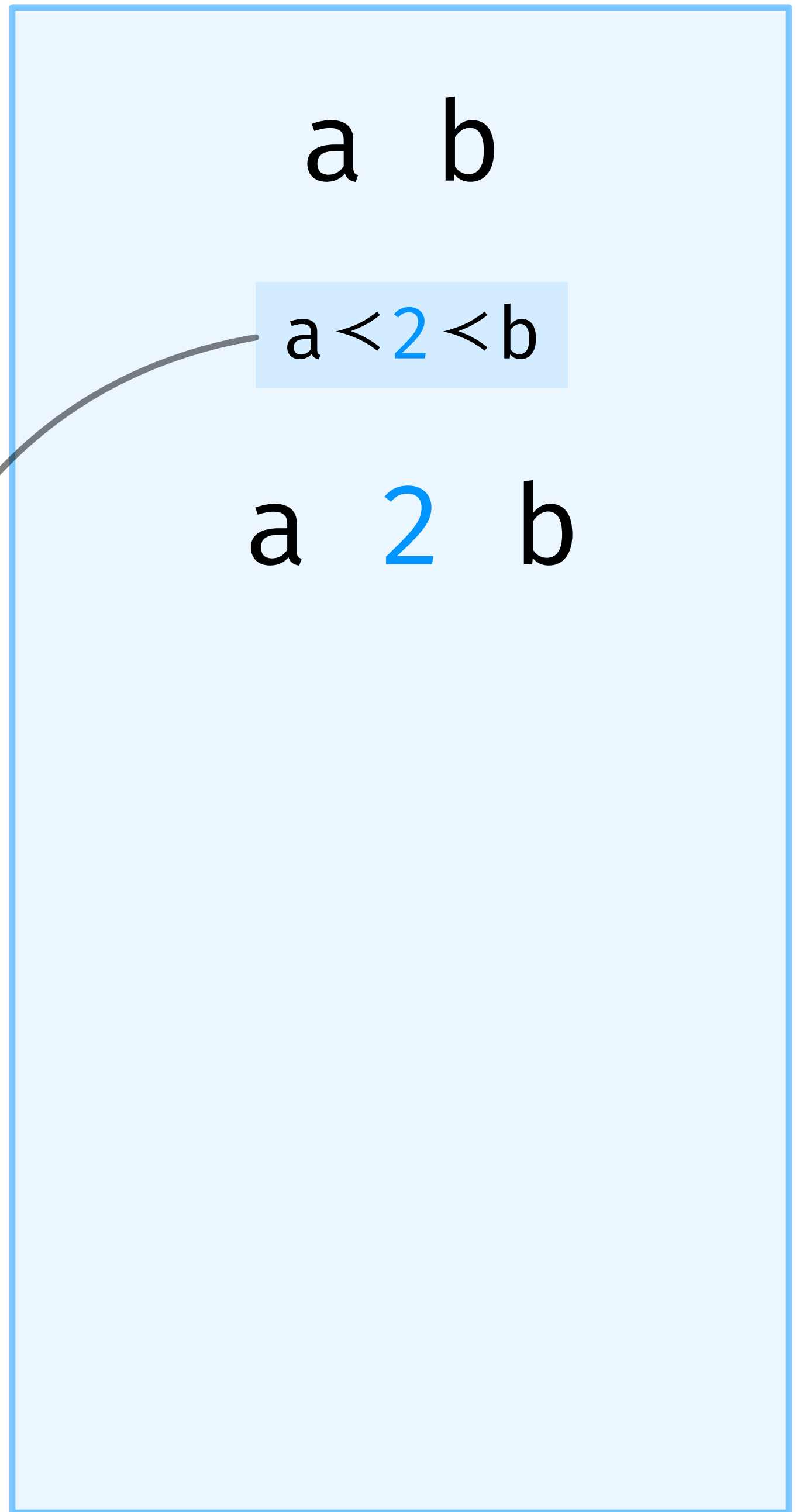
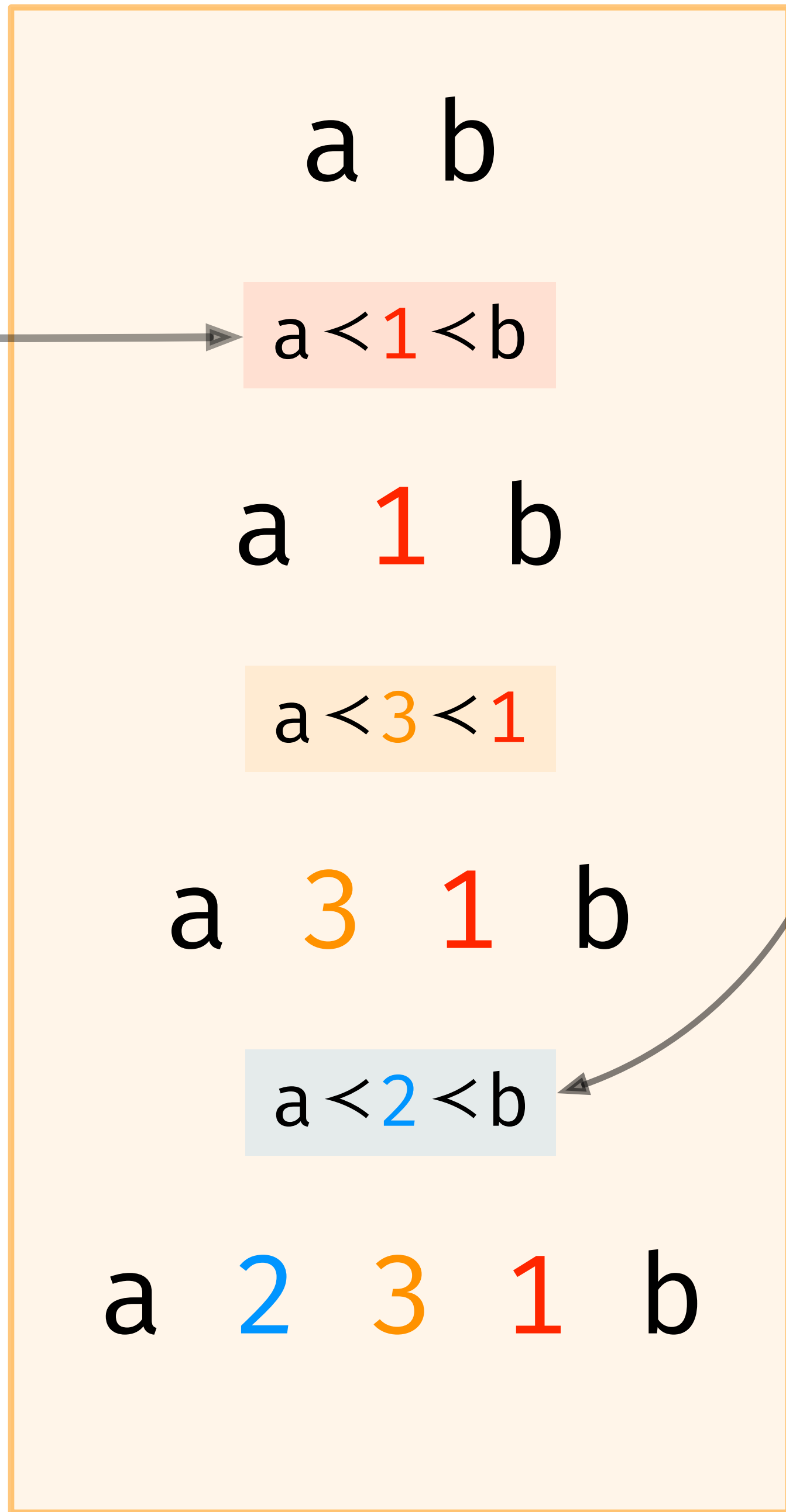
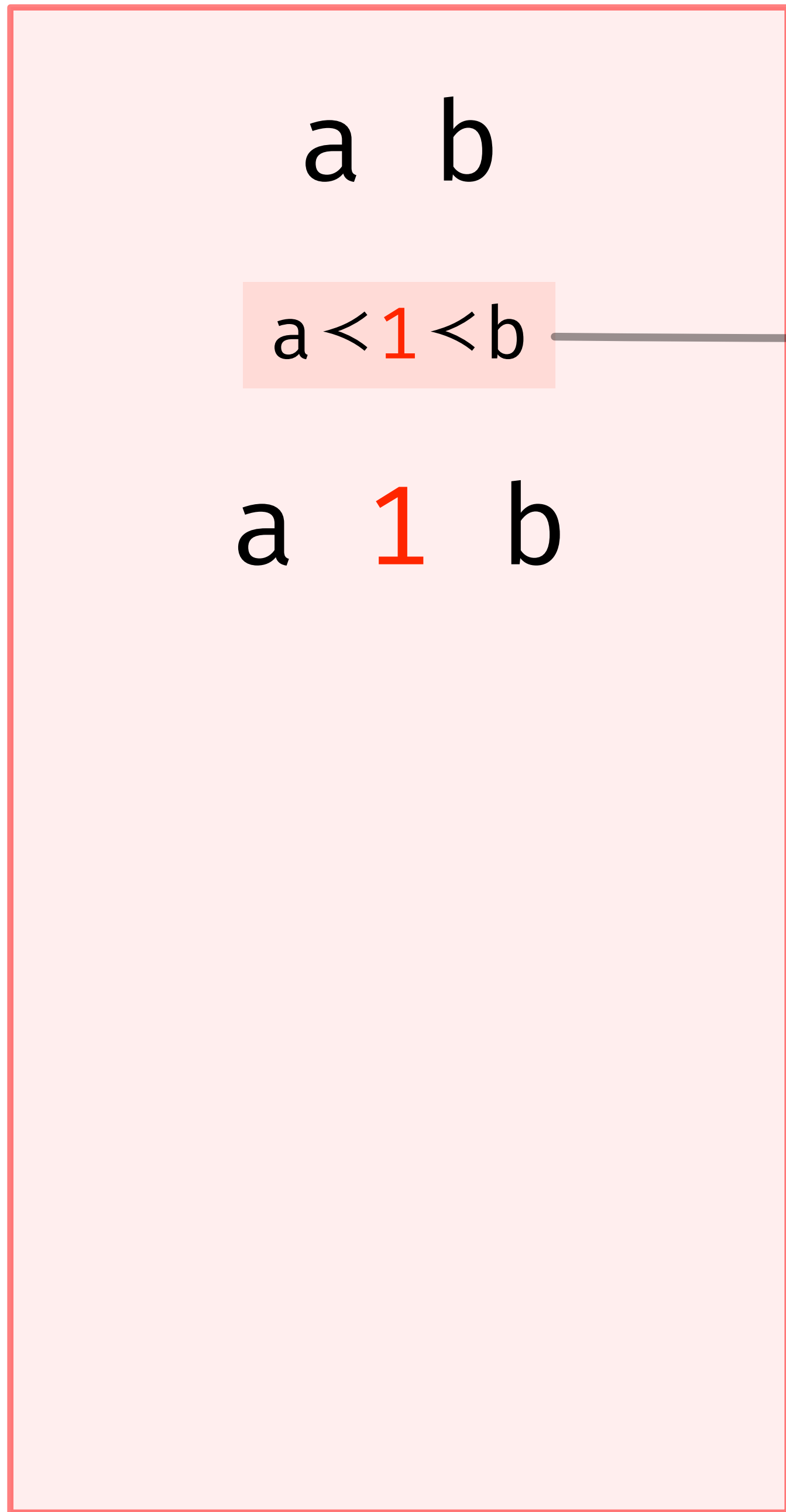
a 2 b

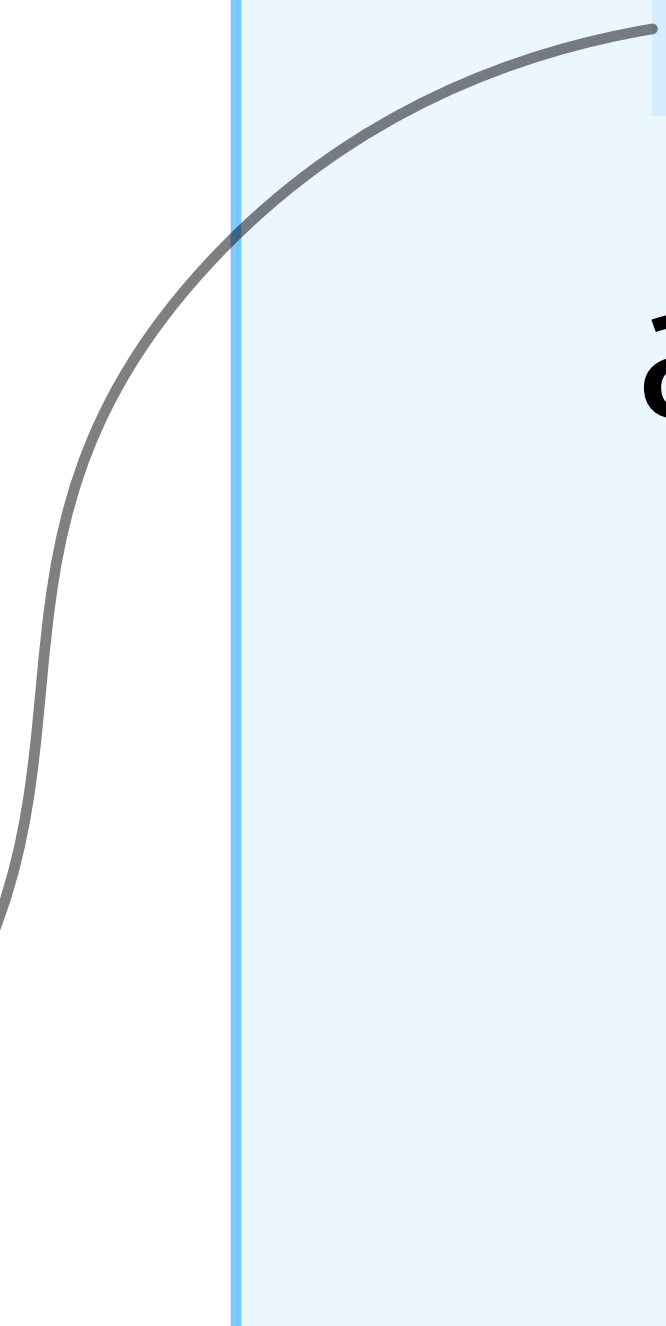
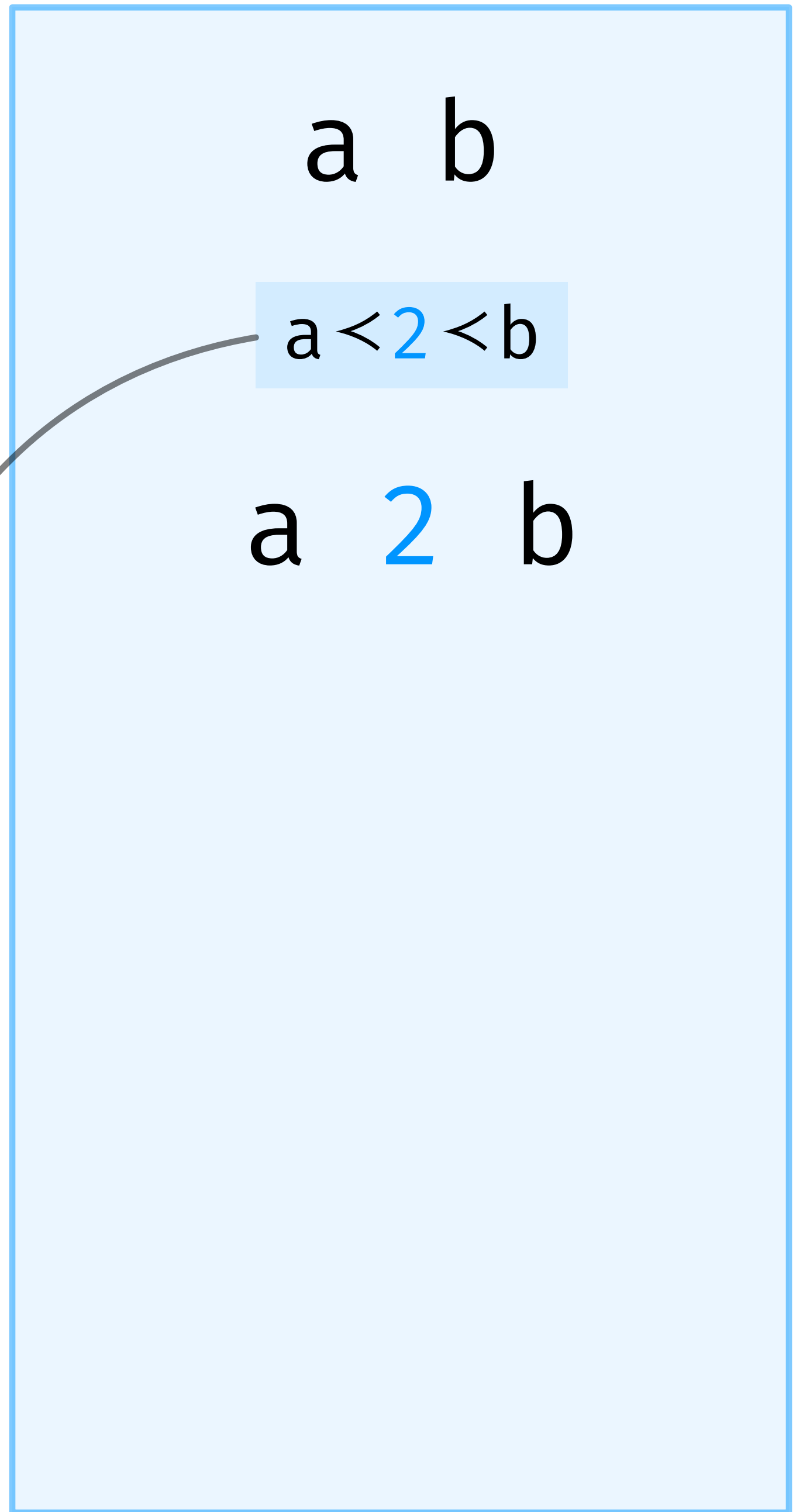
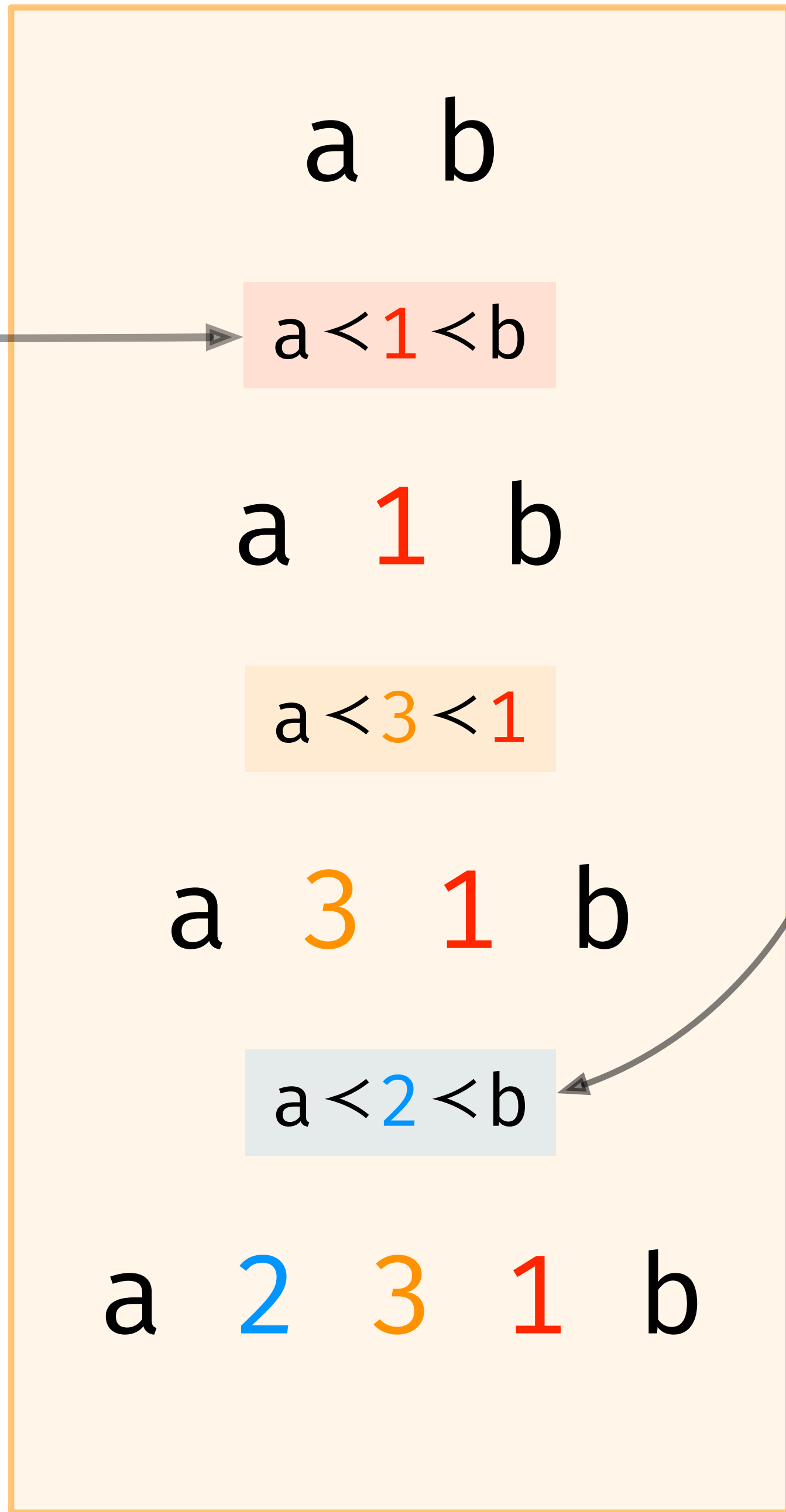
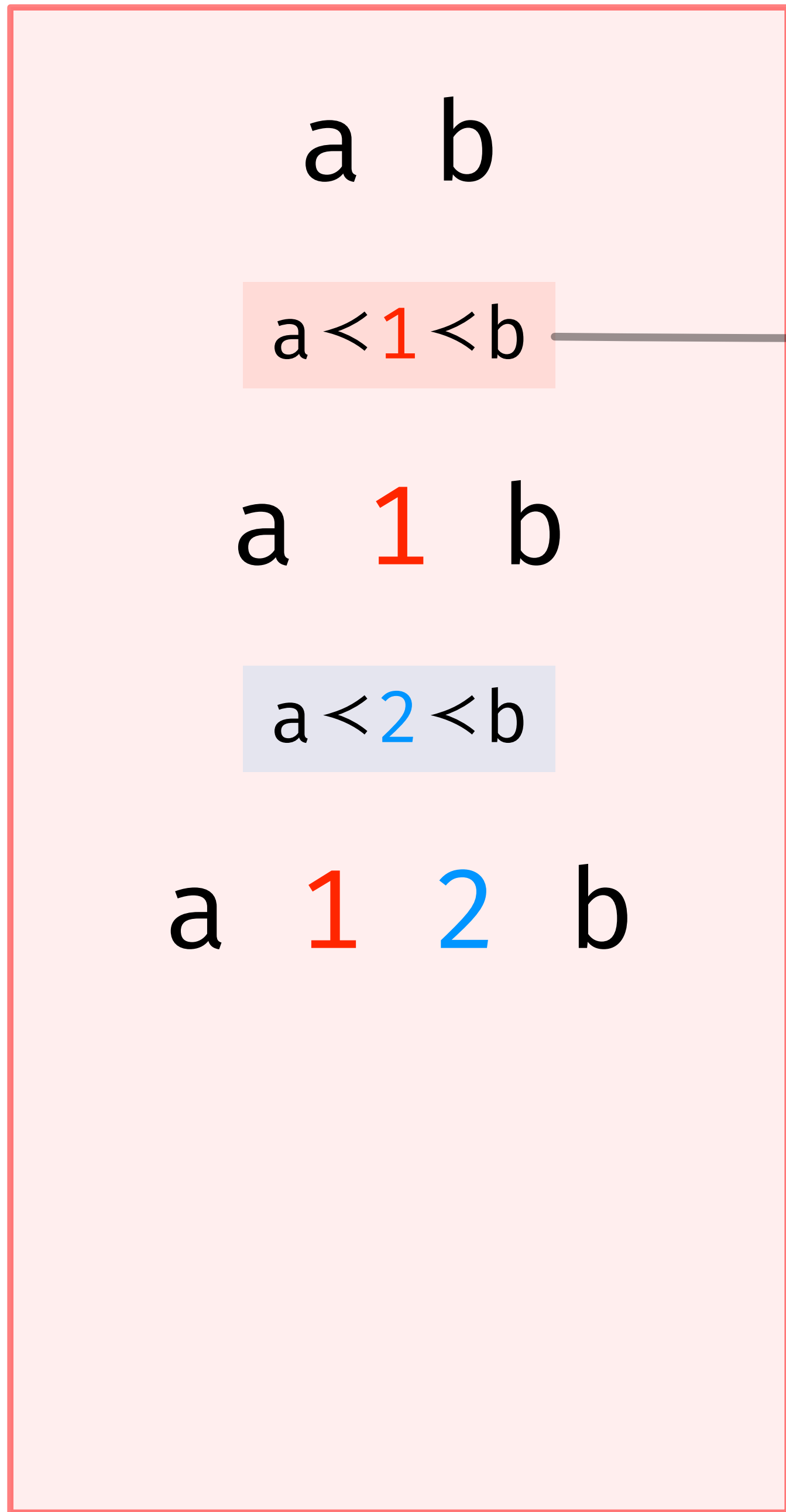


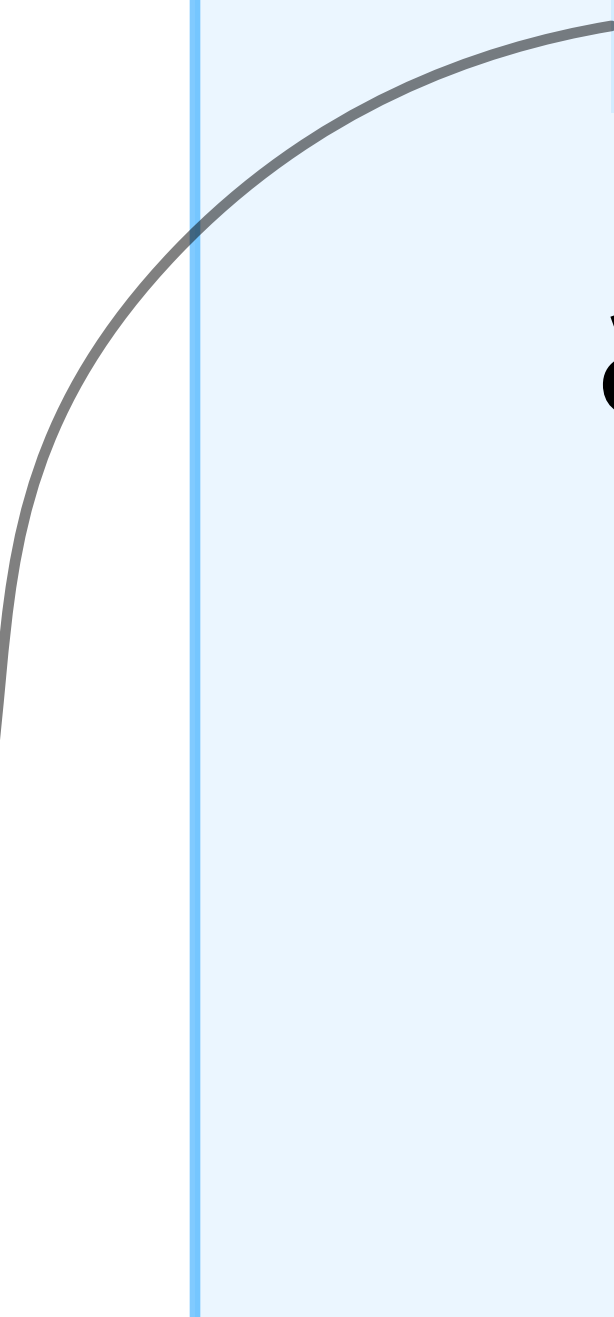
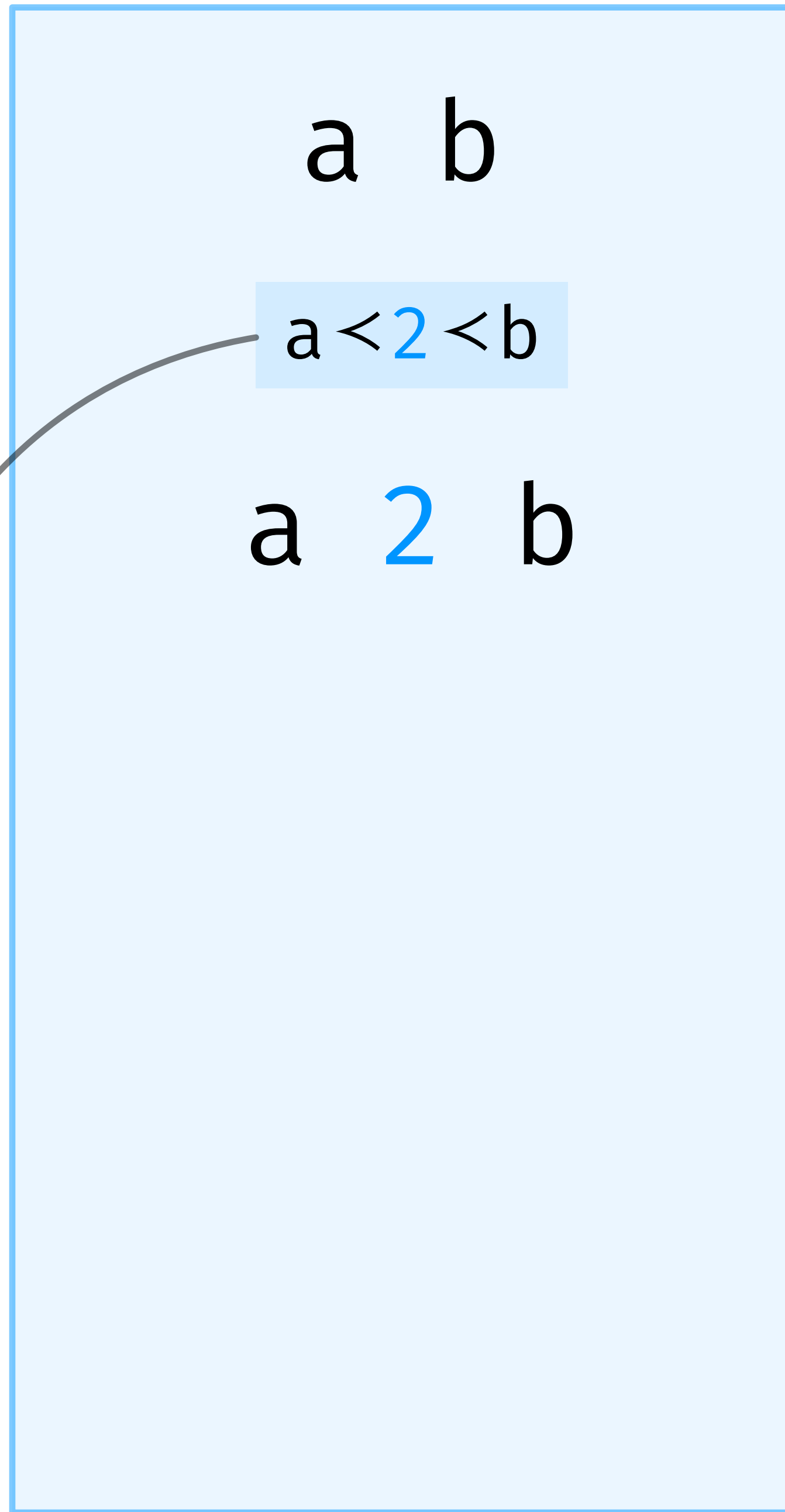
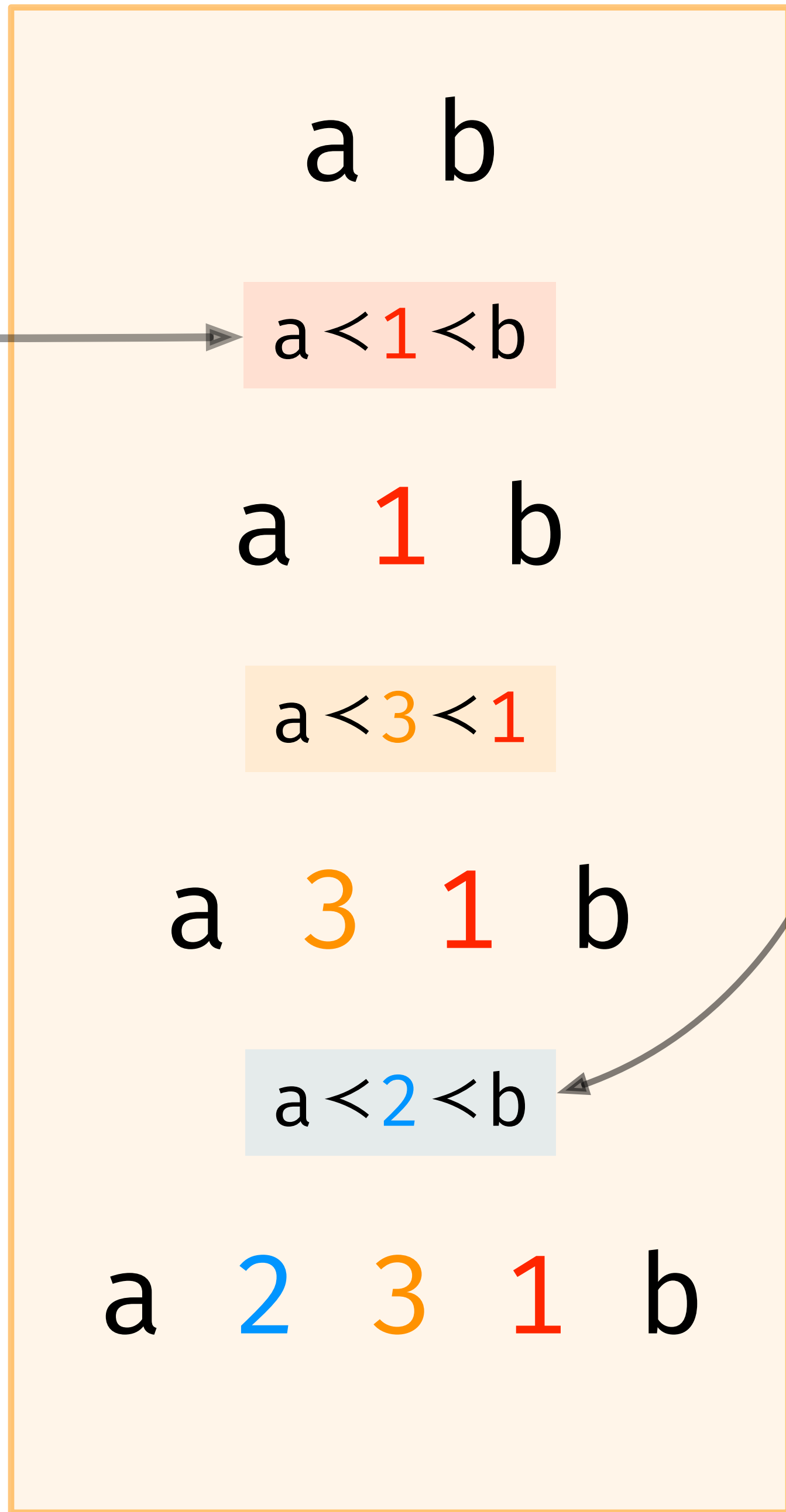
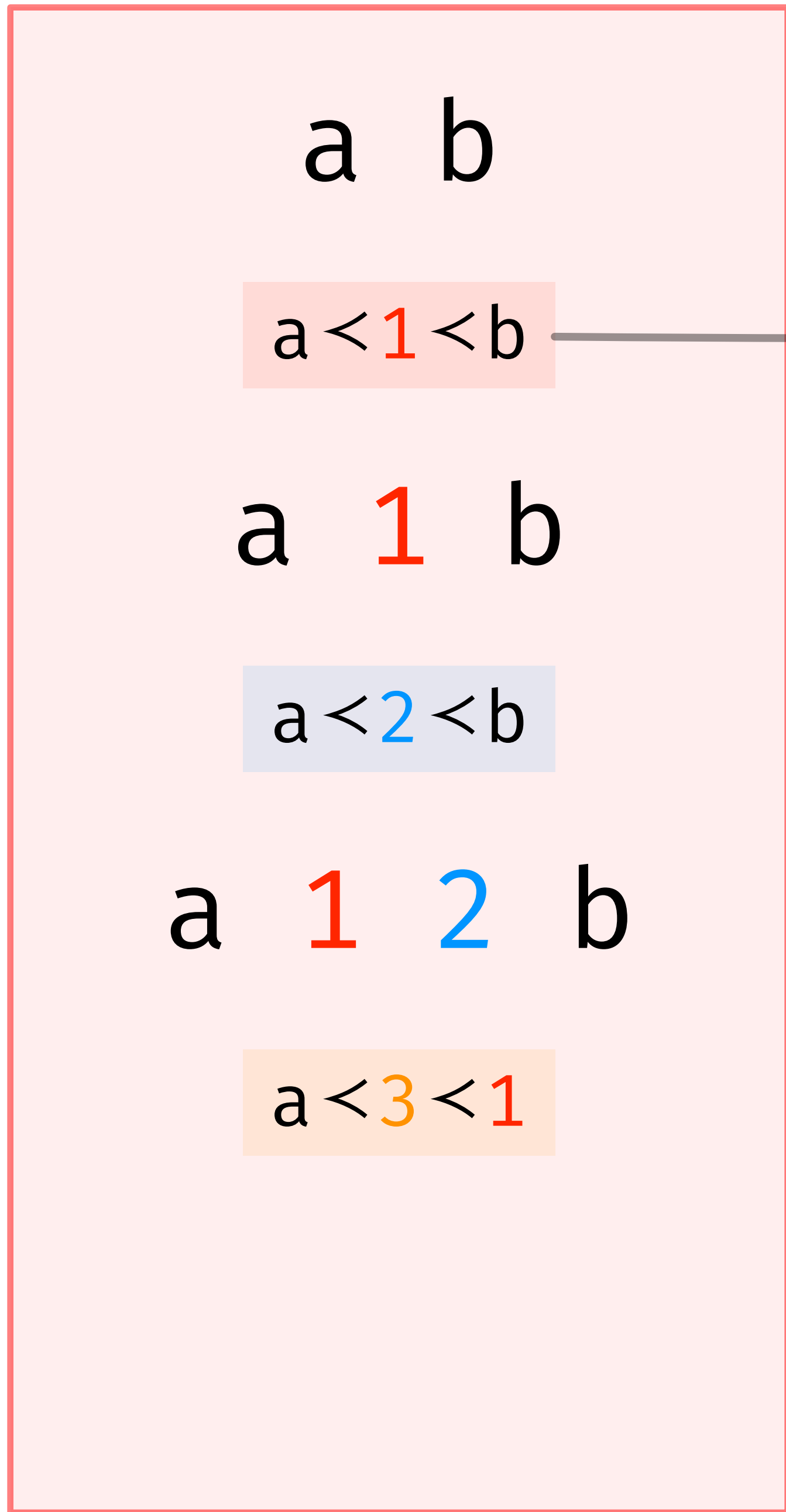


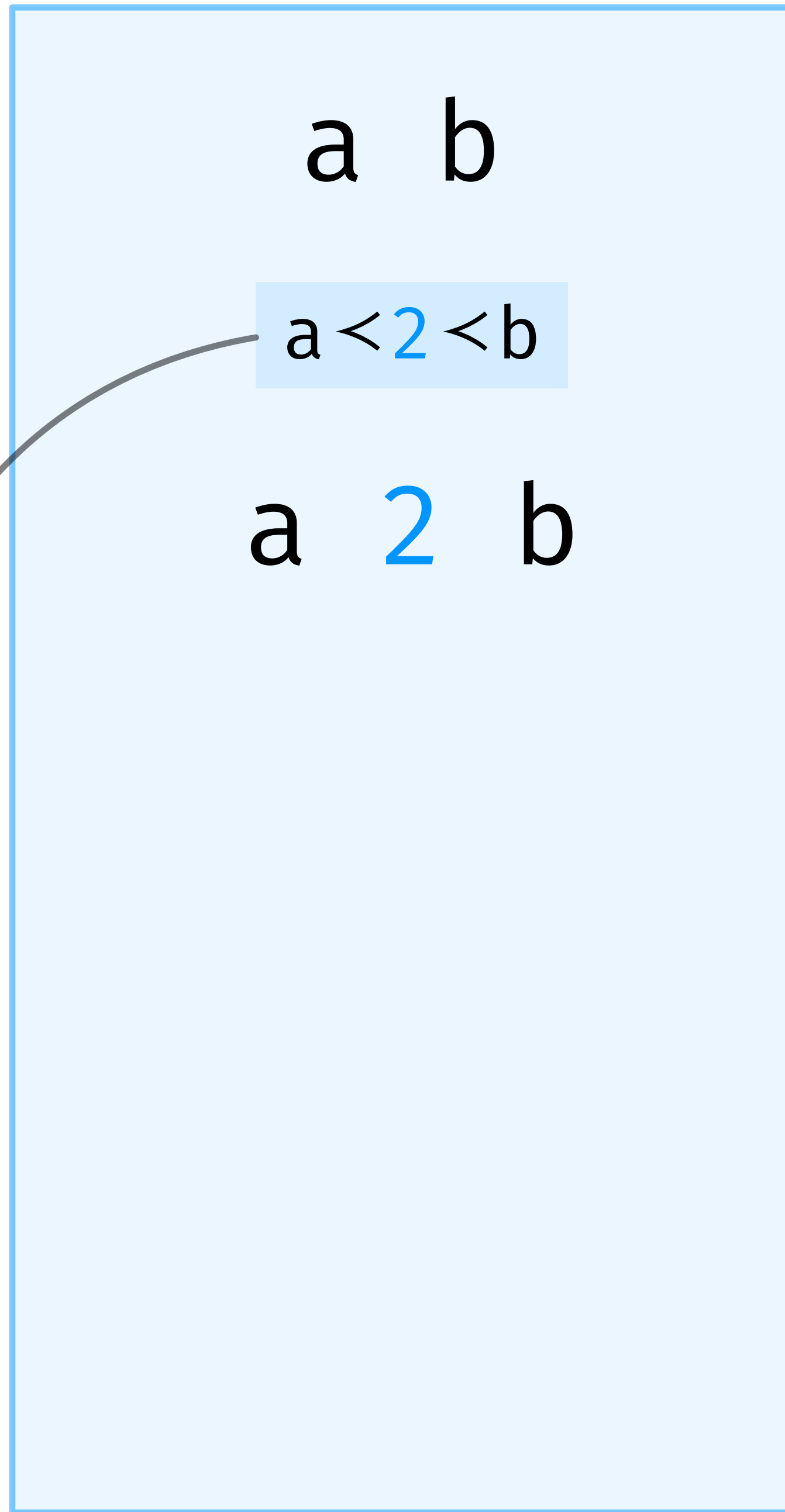
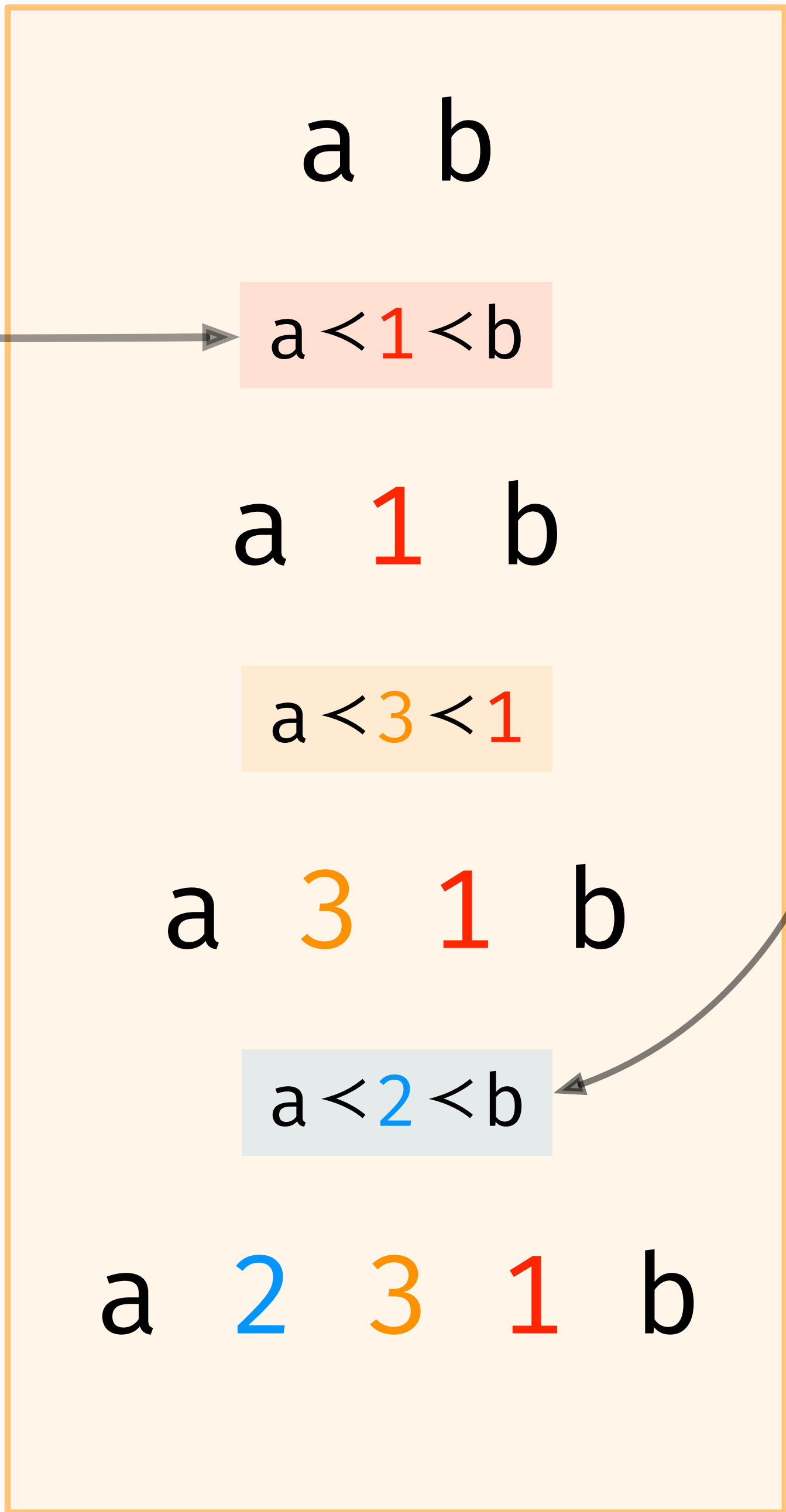
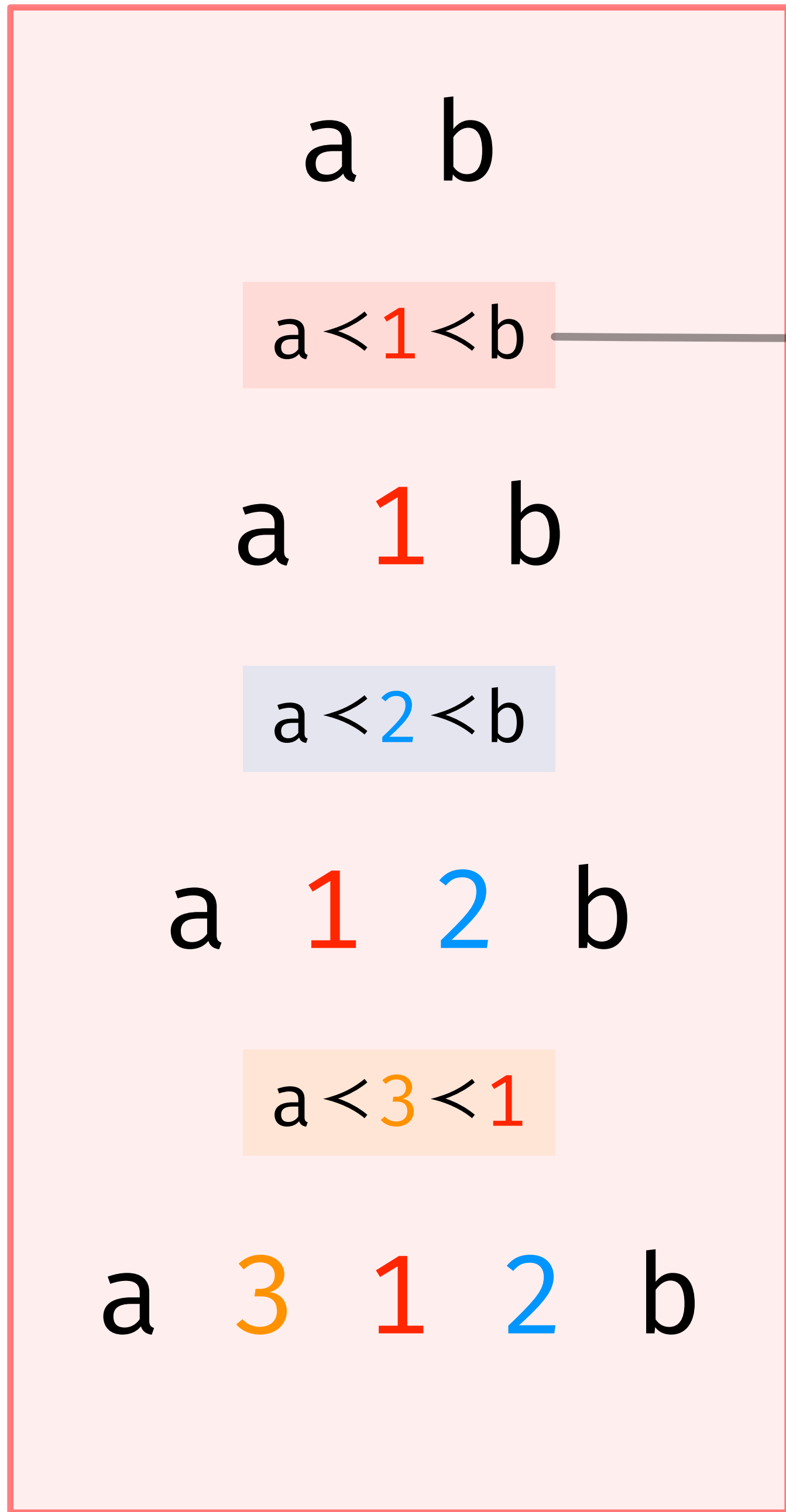


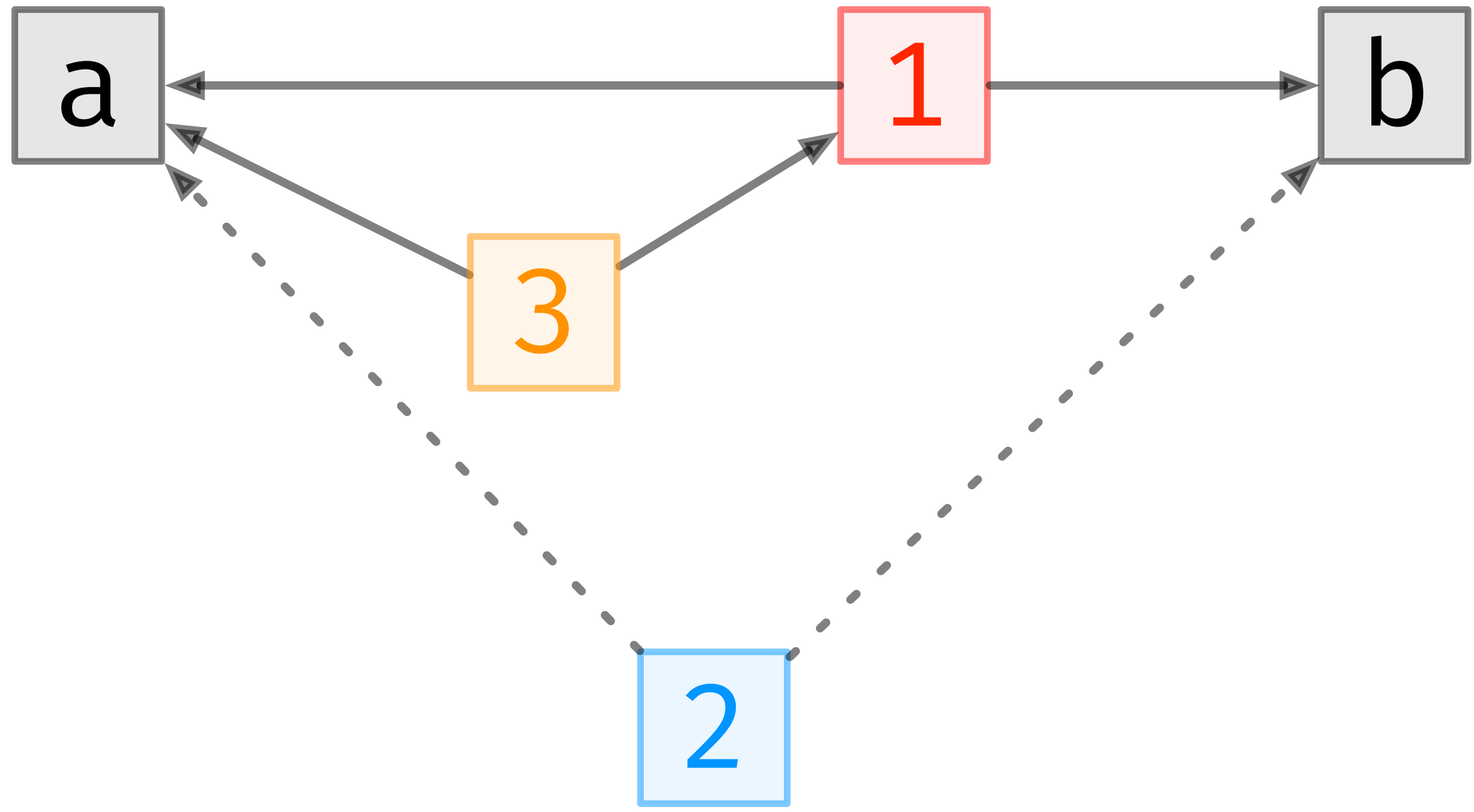


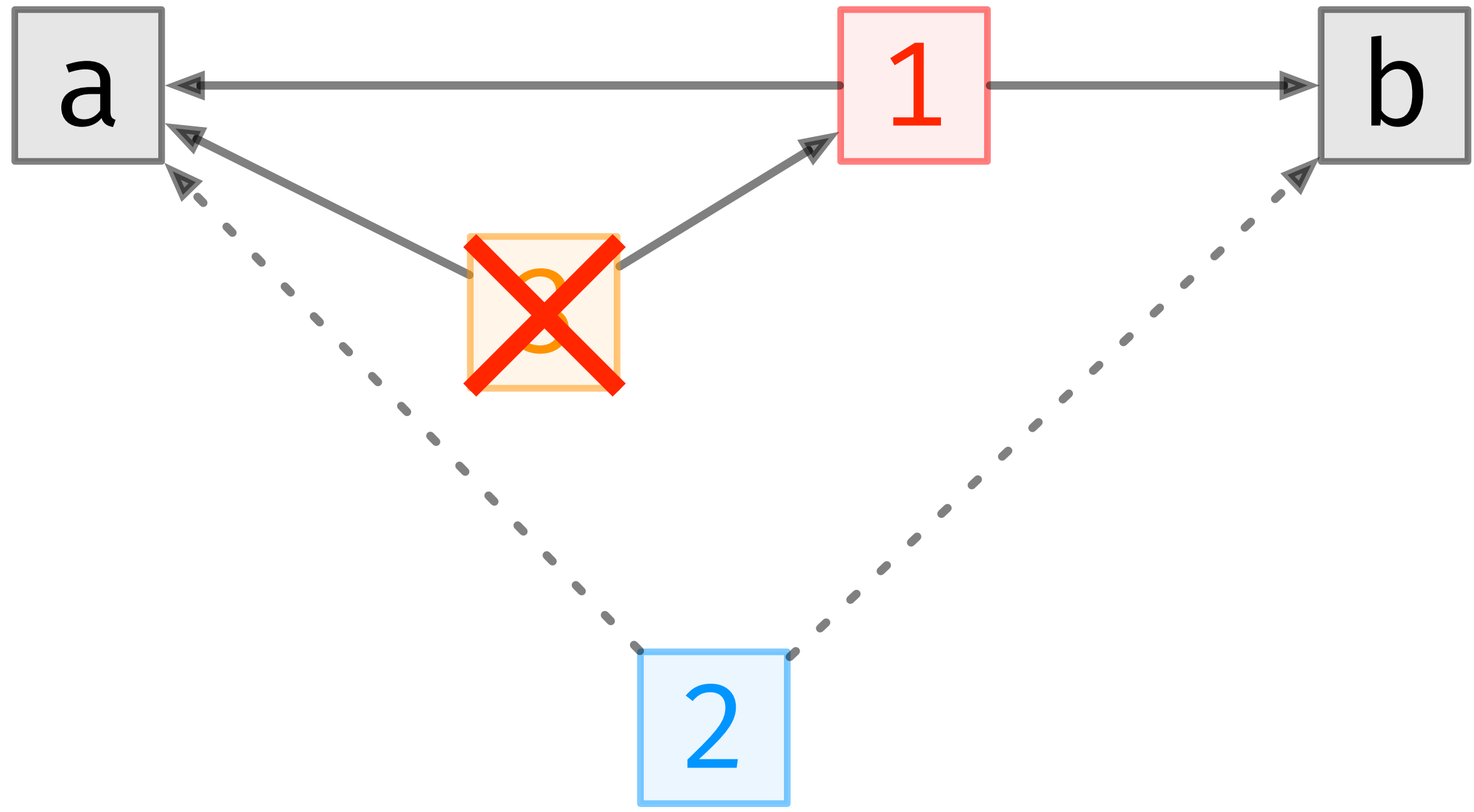


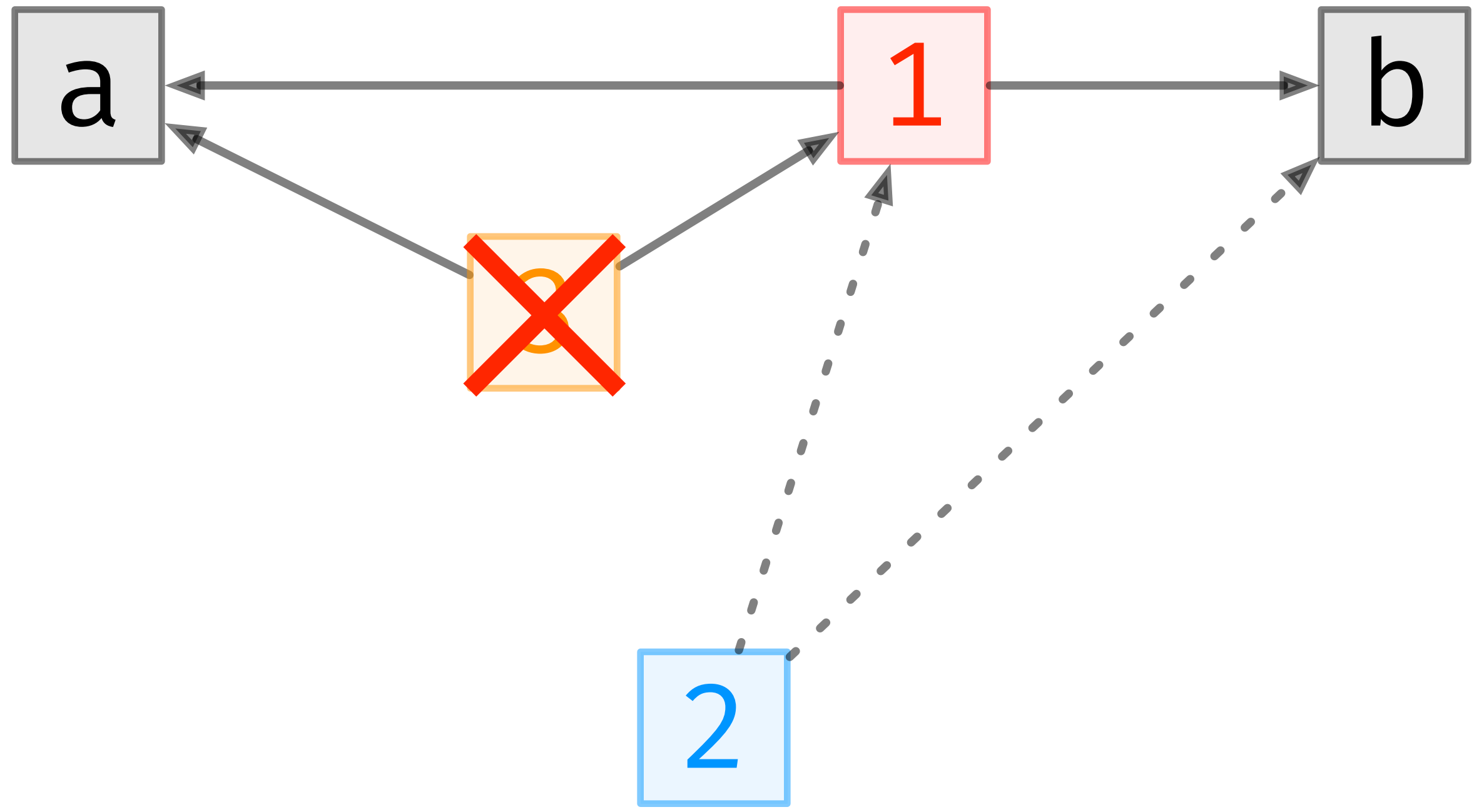












a b

1 a < 1 < b

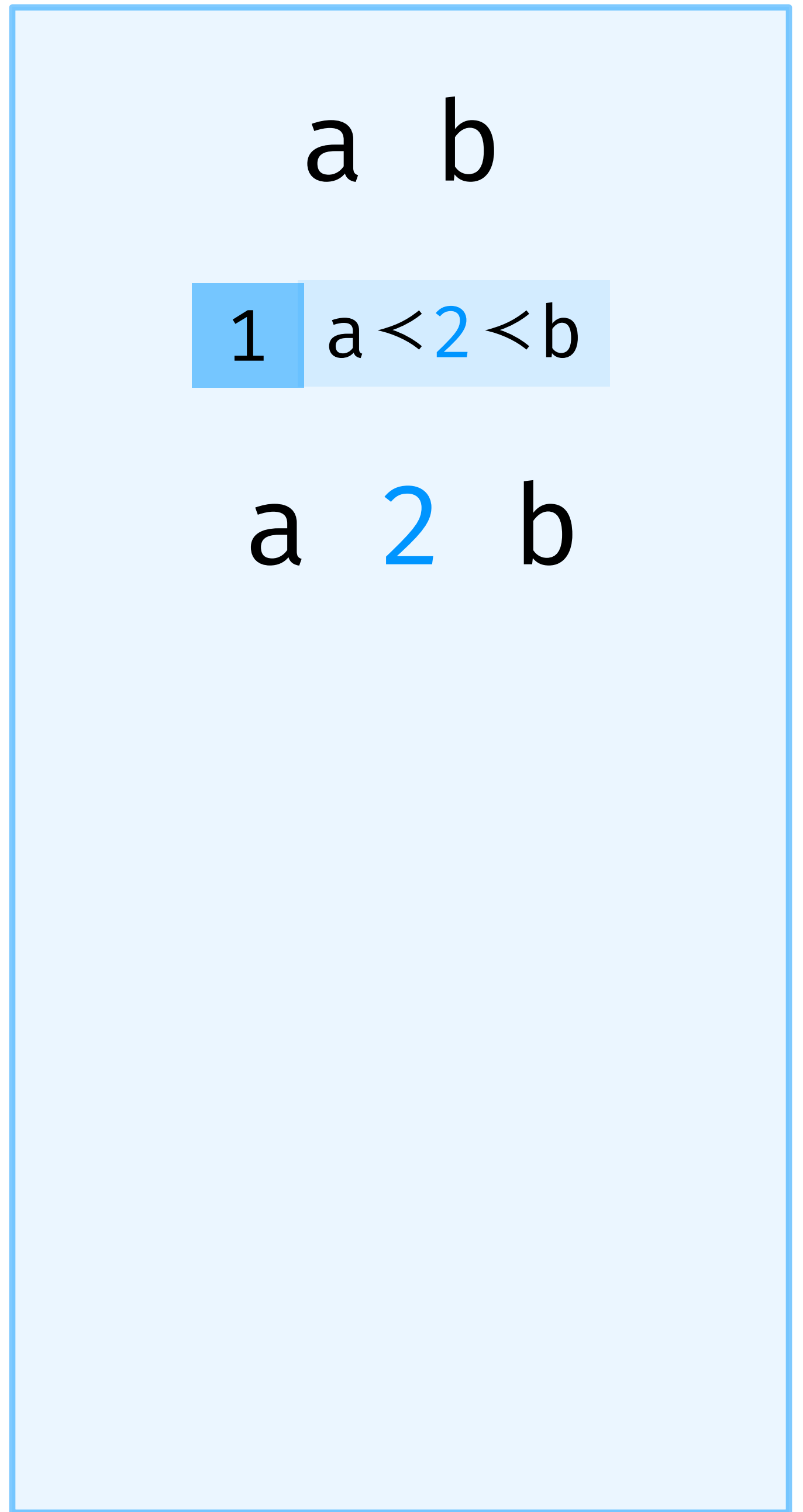
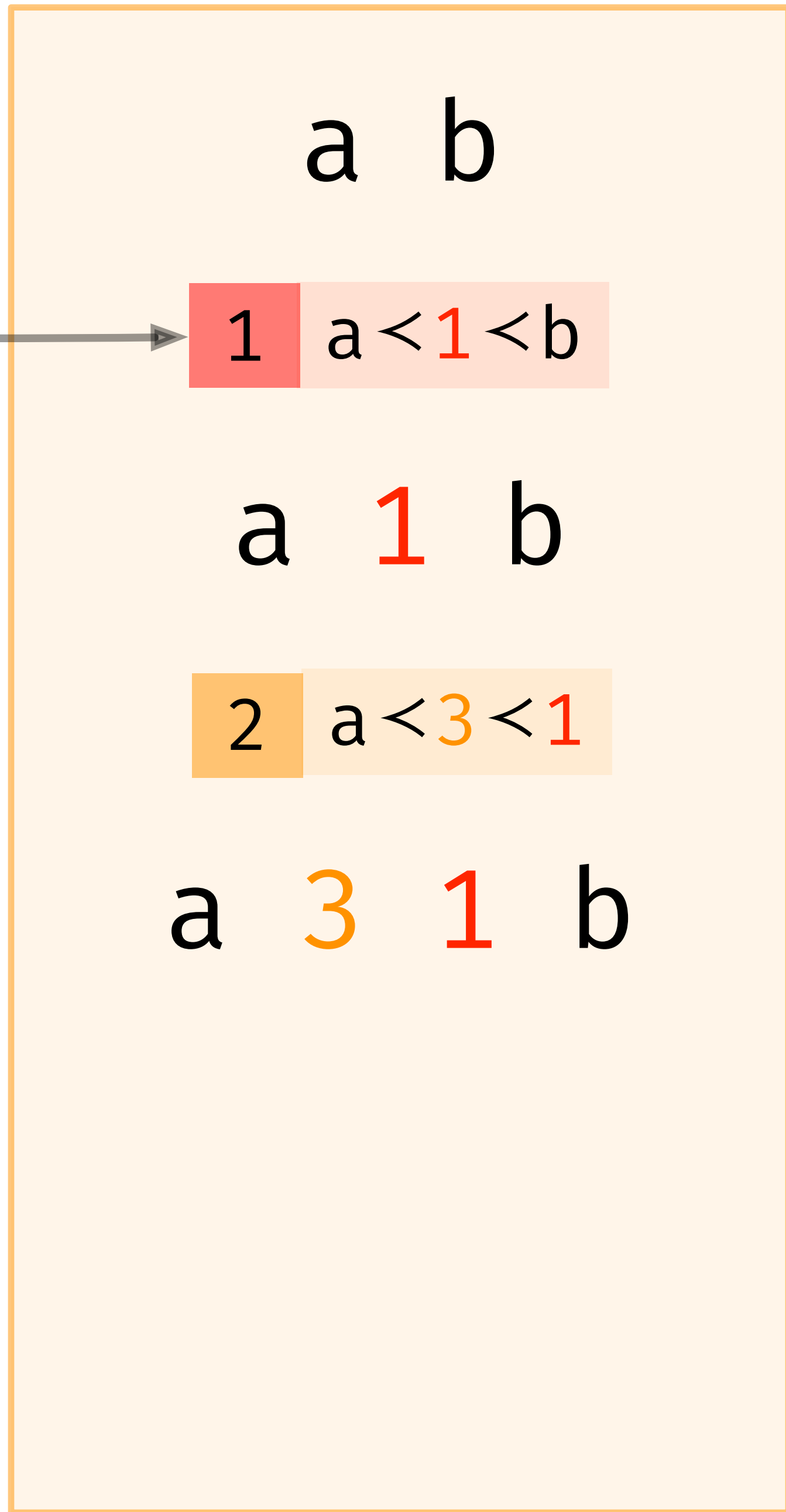
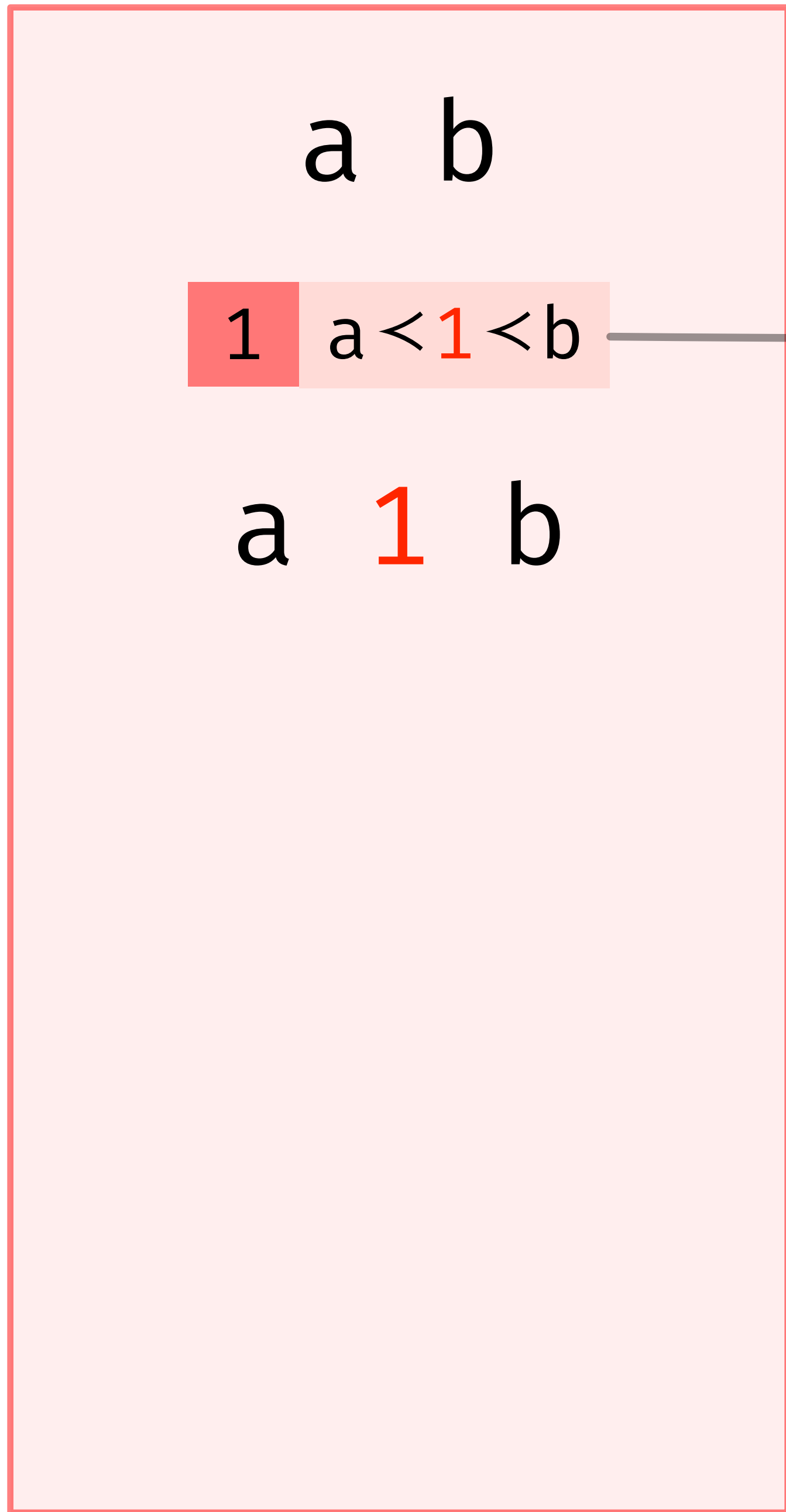
a 1 b

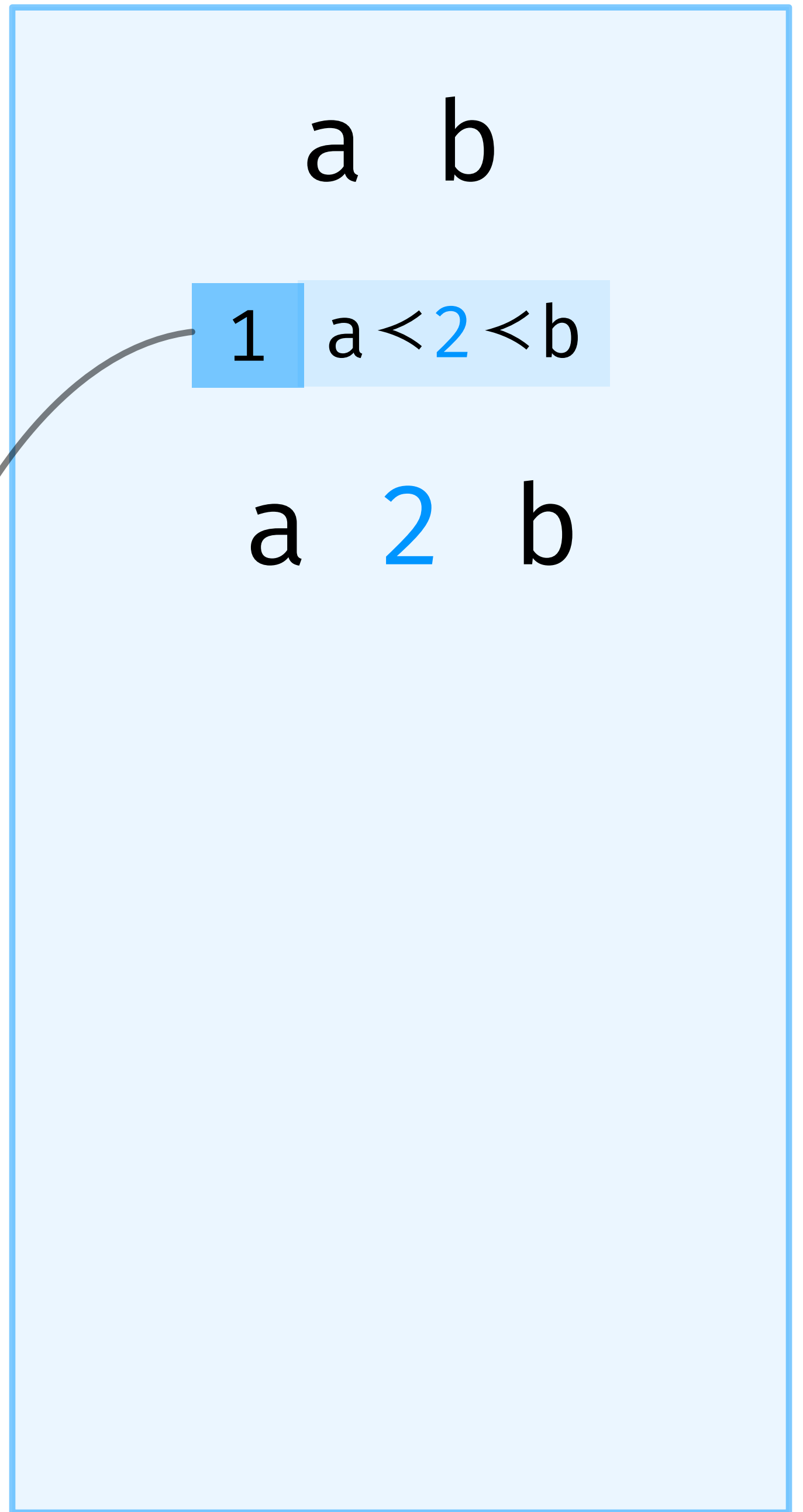
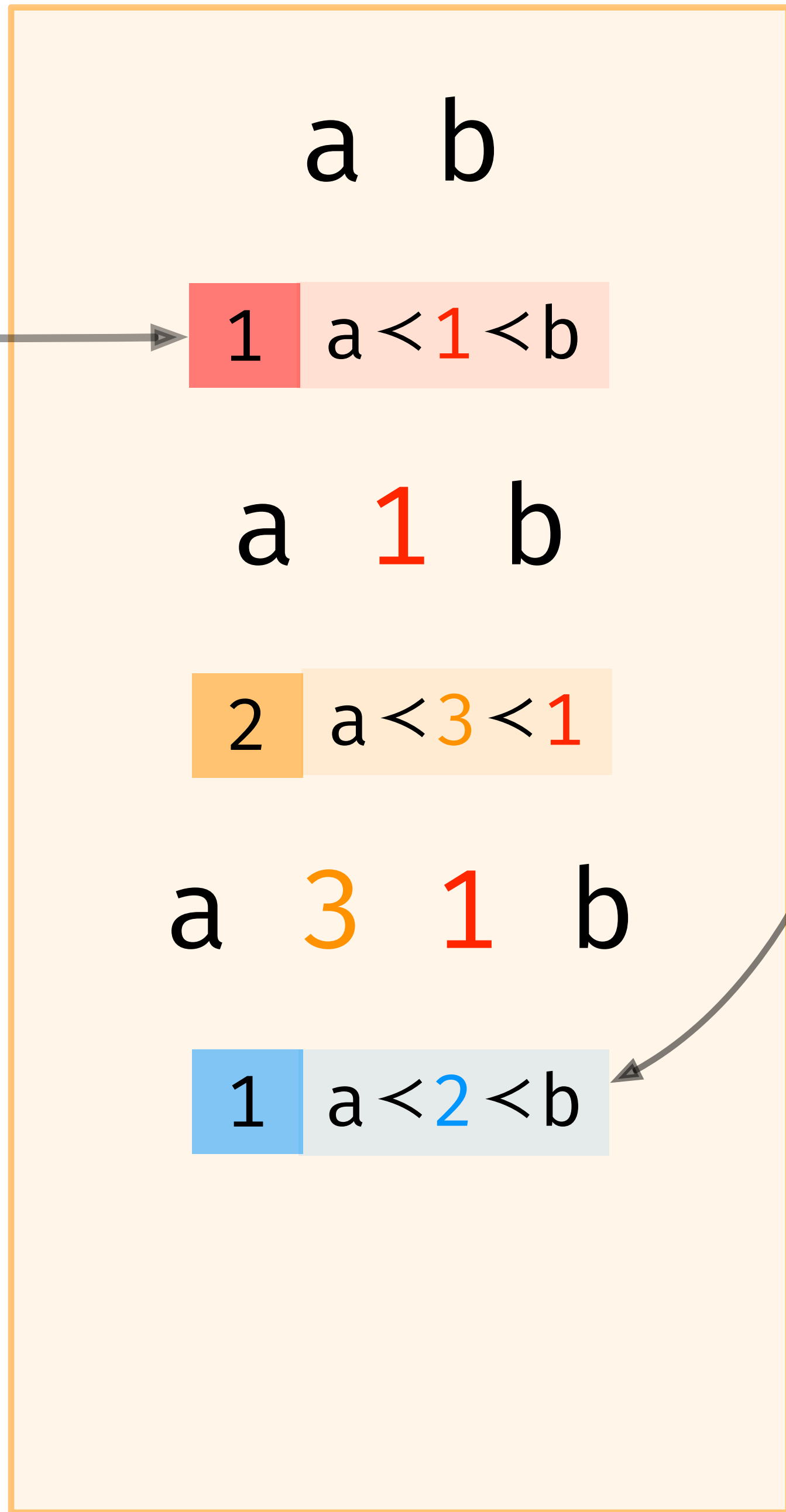
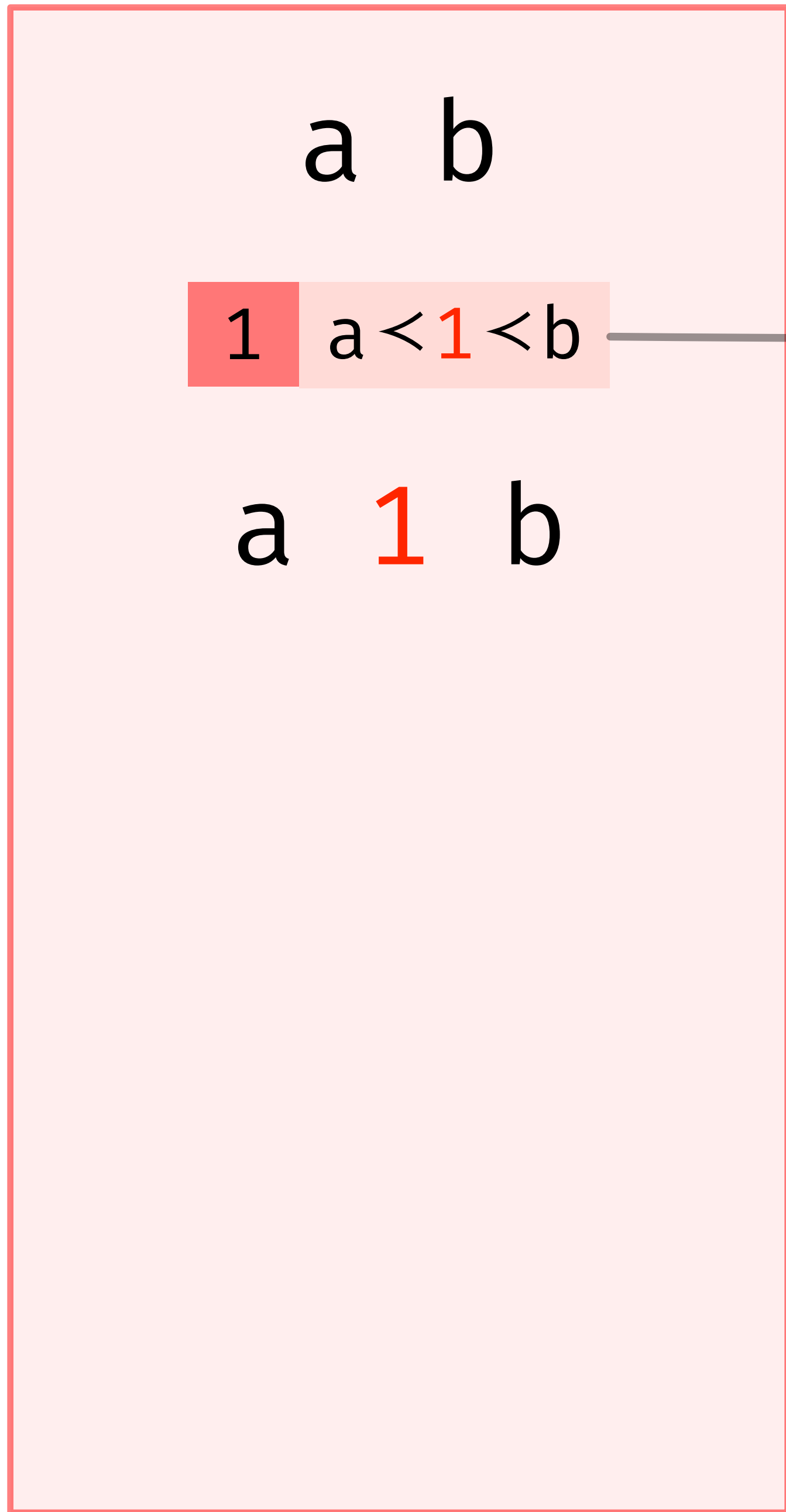
a b

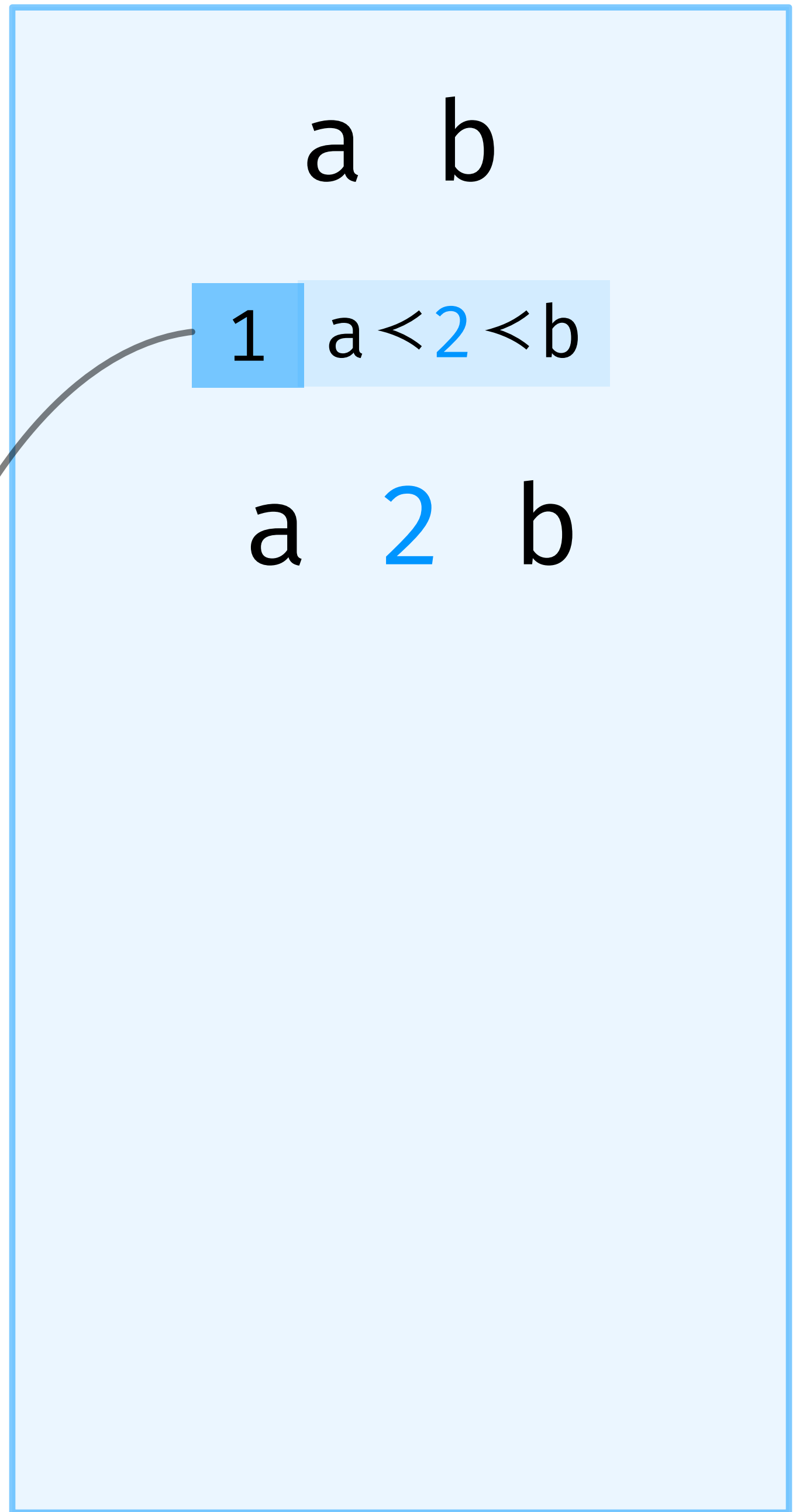
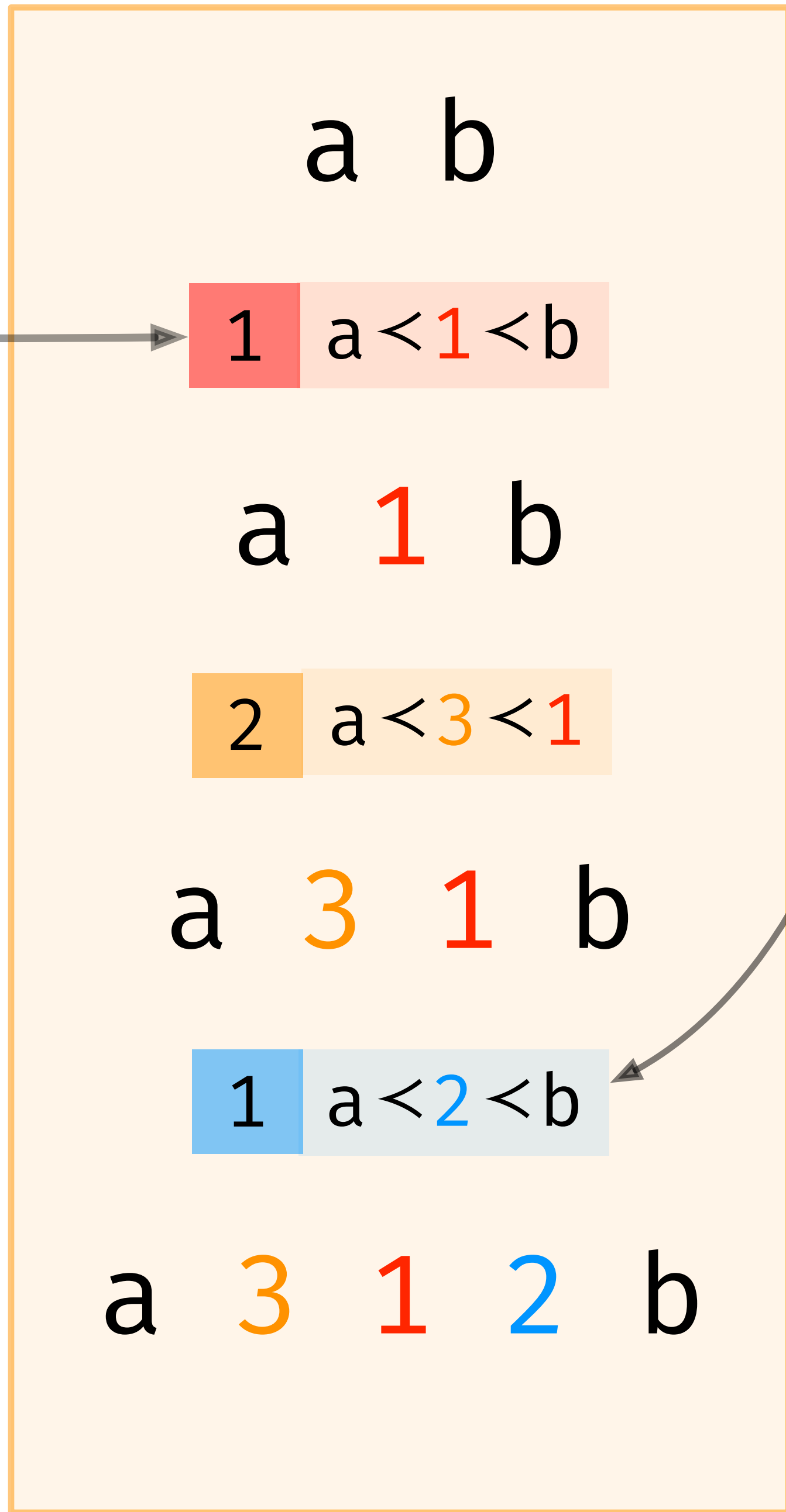
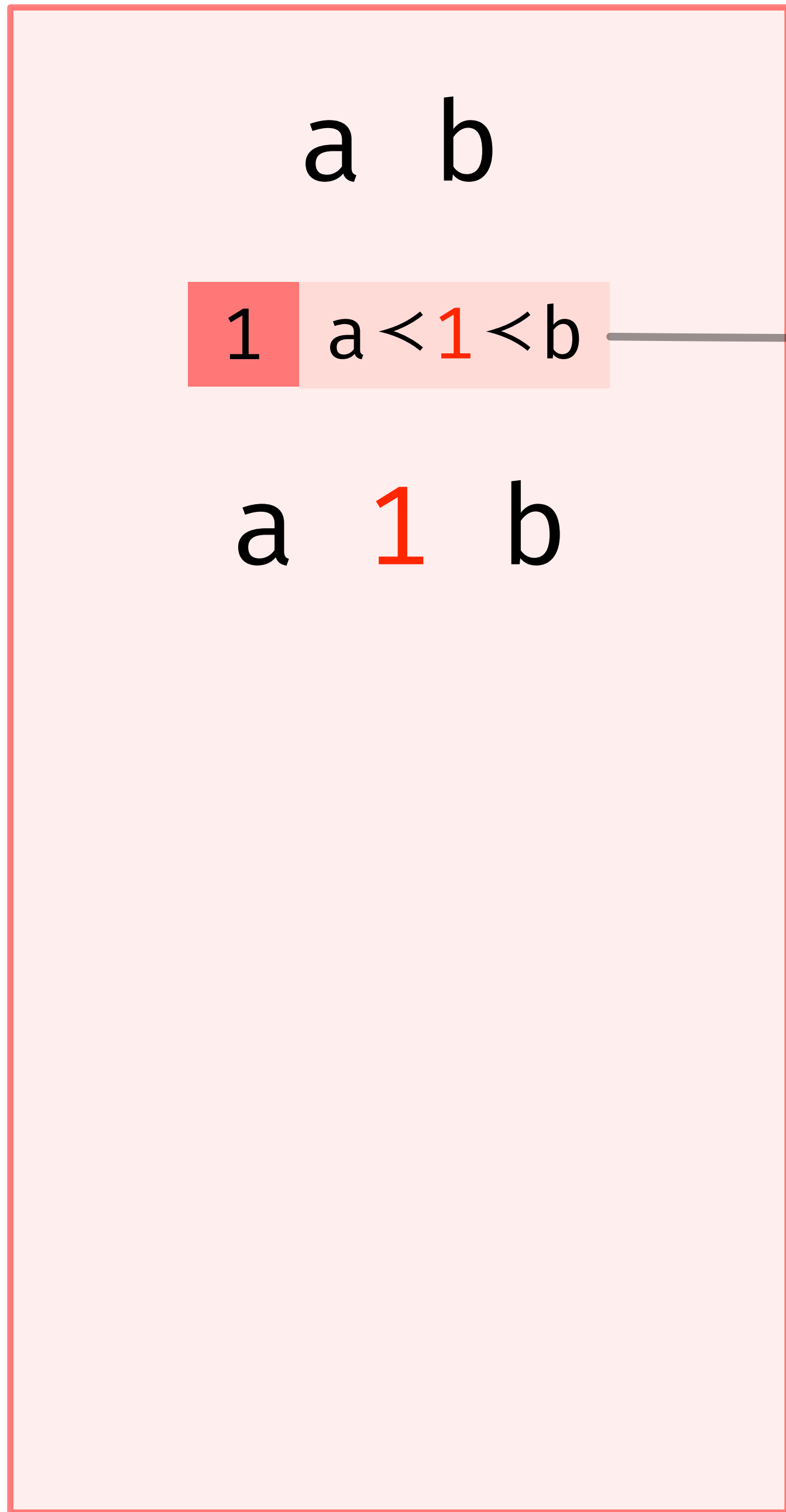
a b

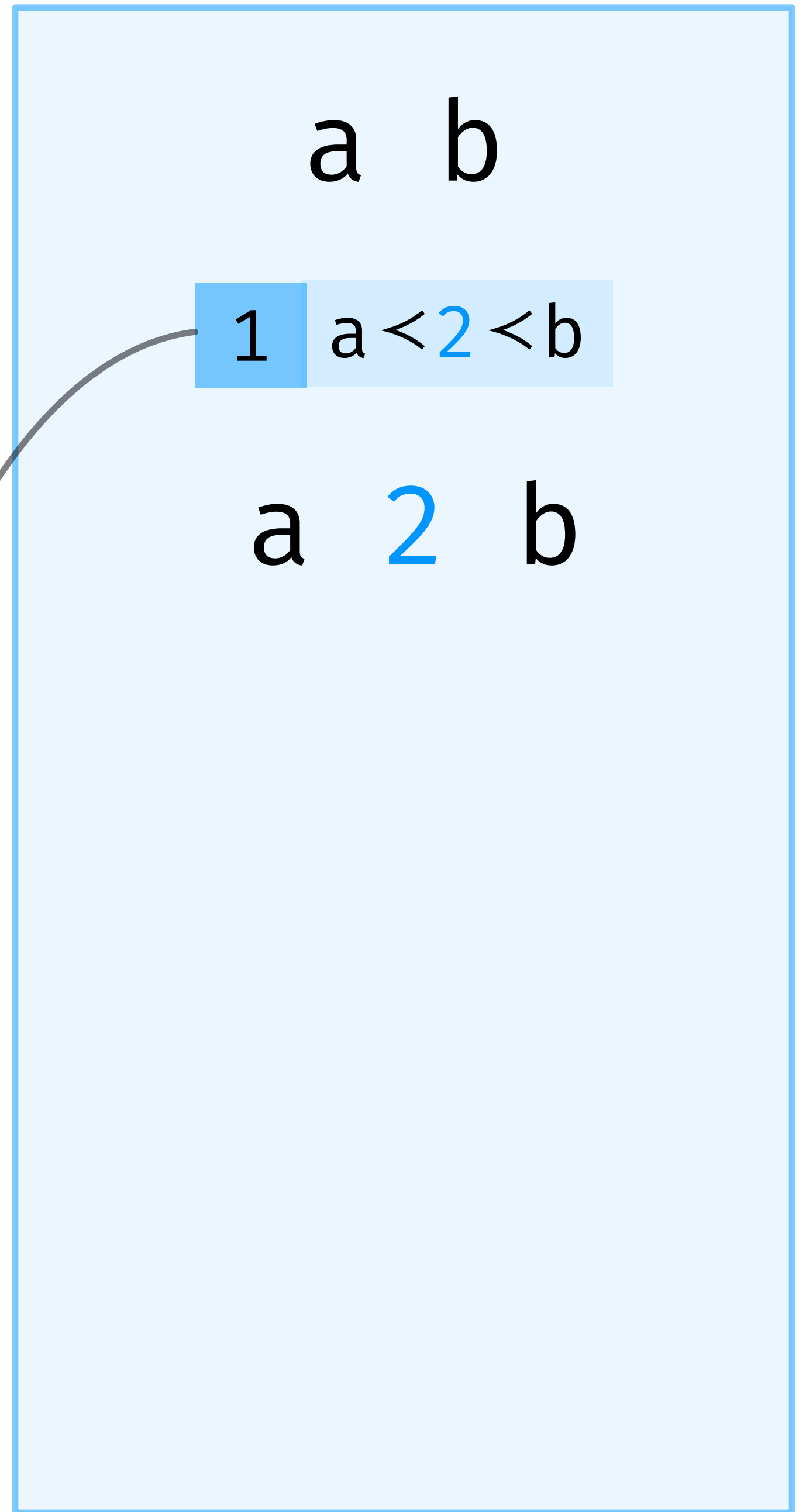
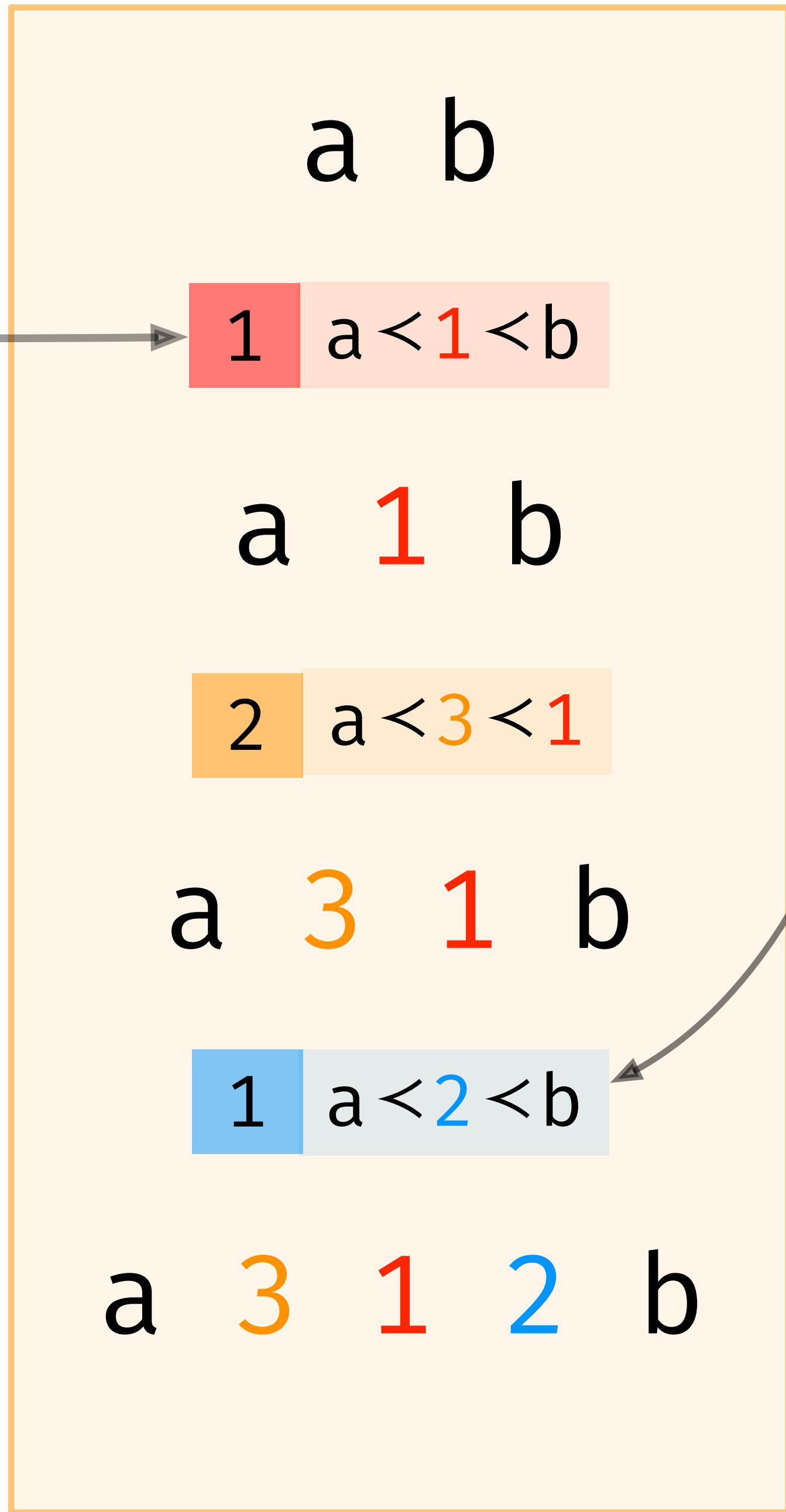
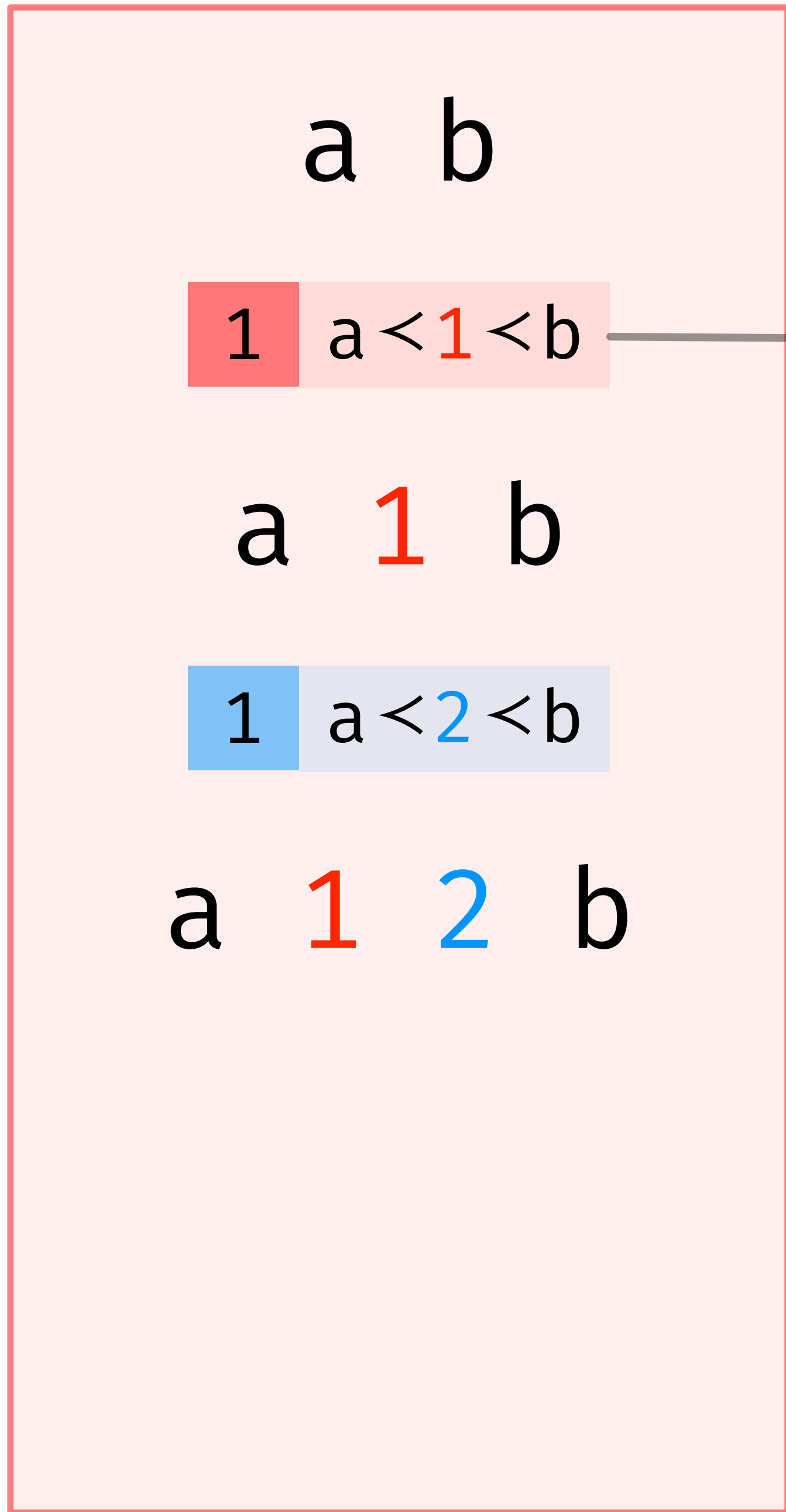
1 a < 2 < b

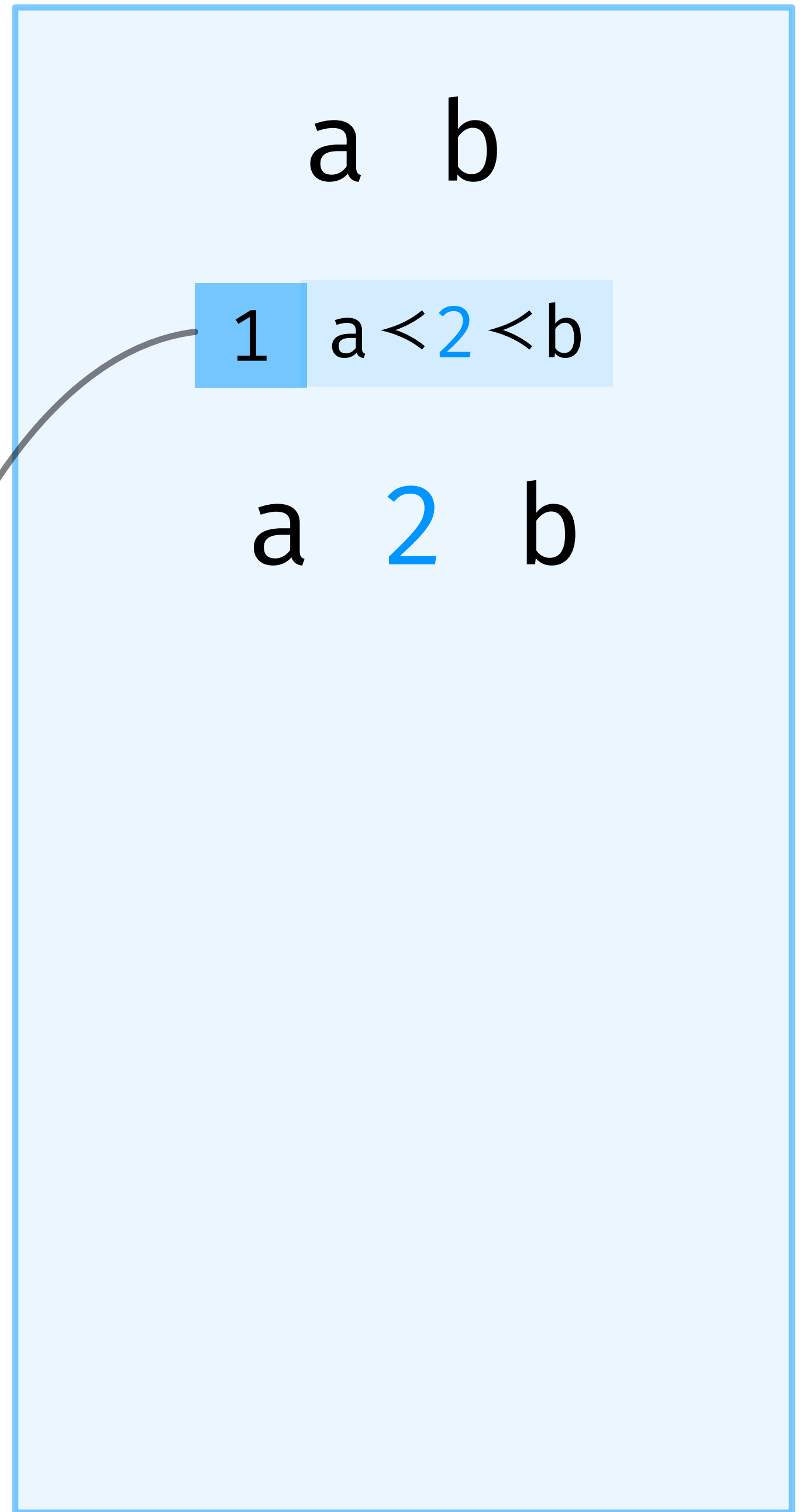
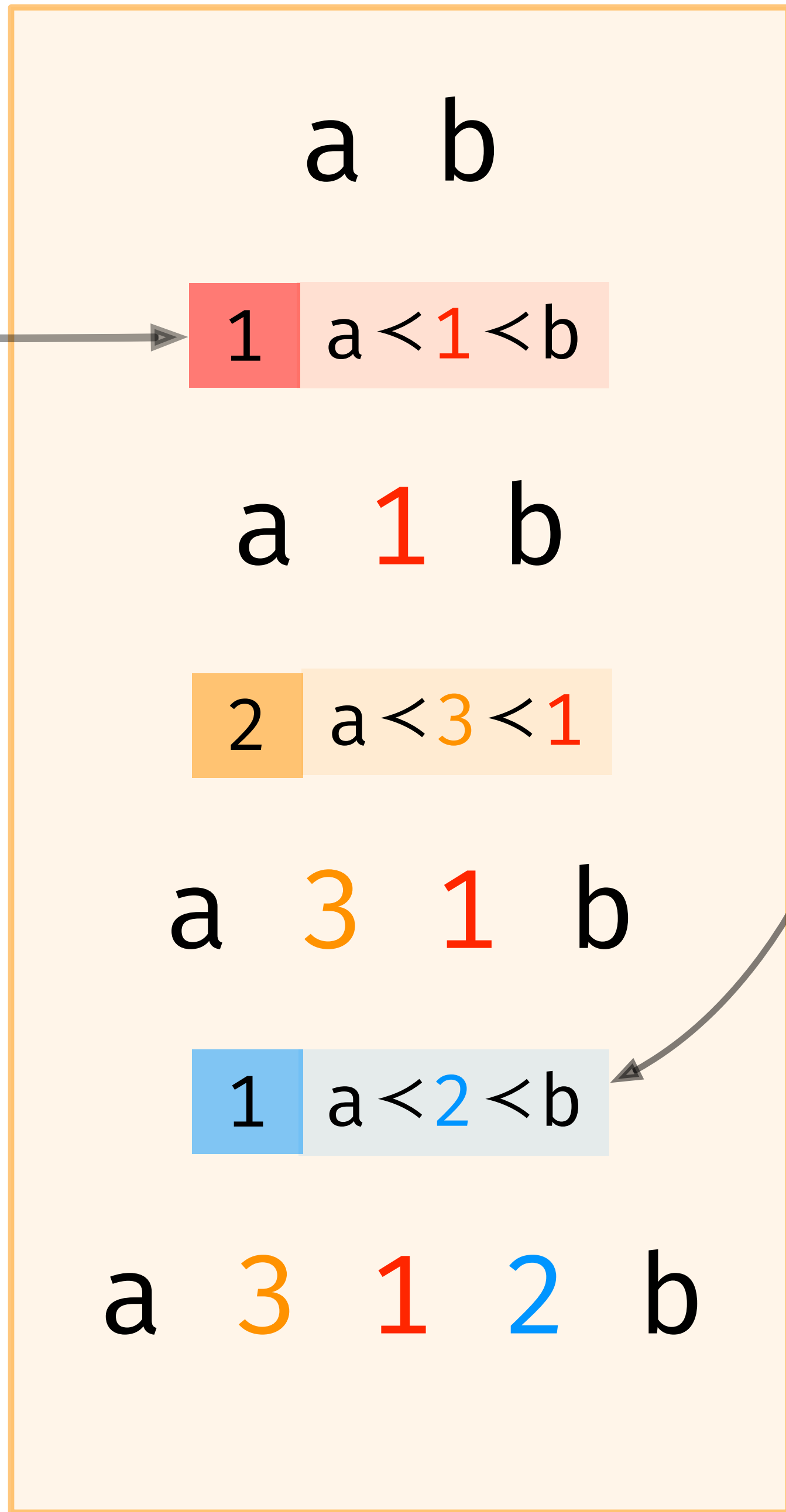
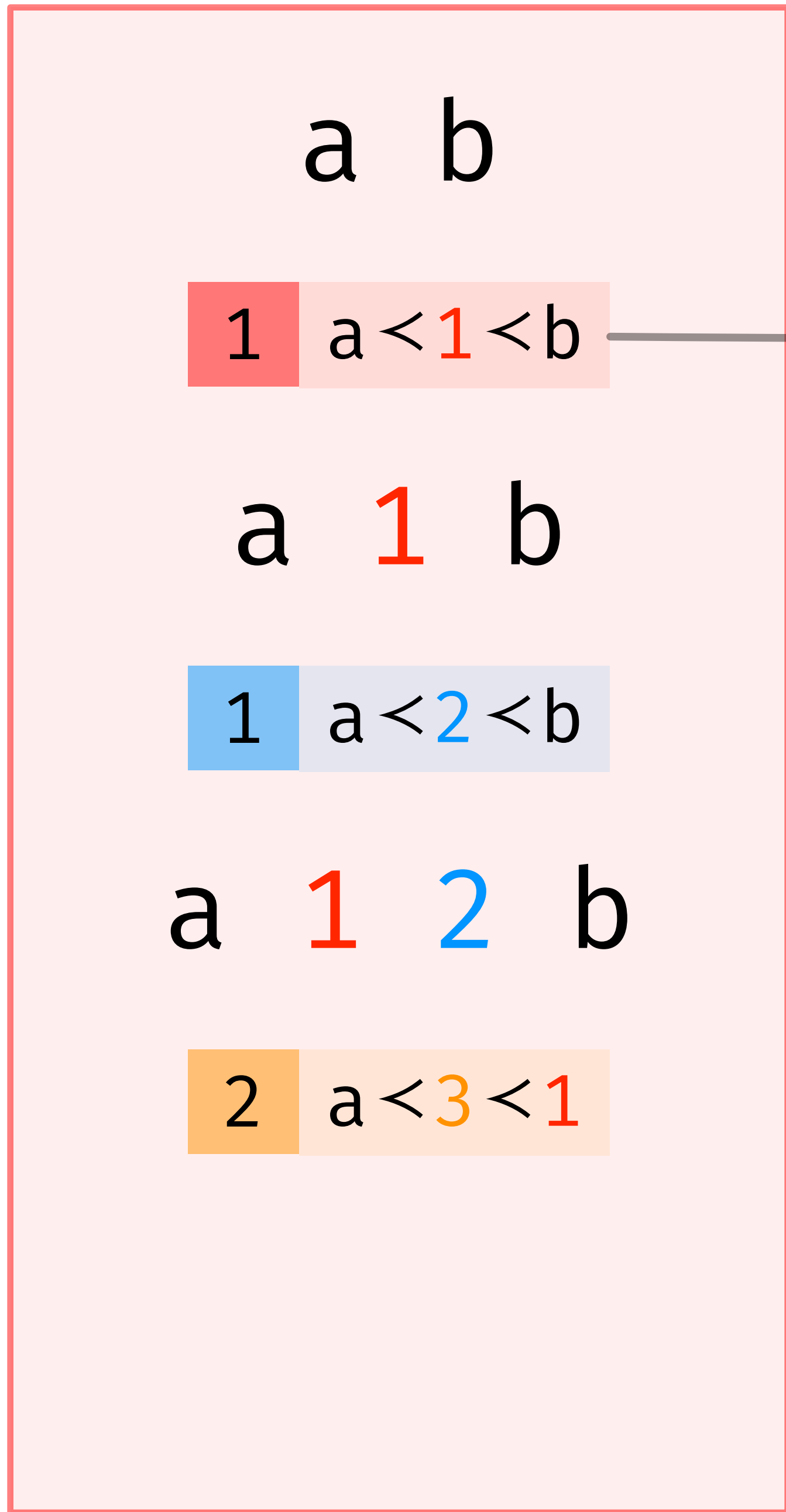
a 2 b

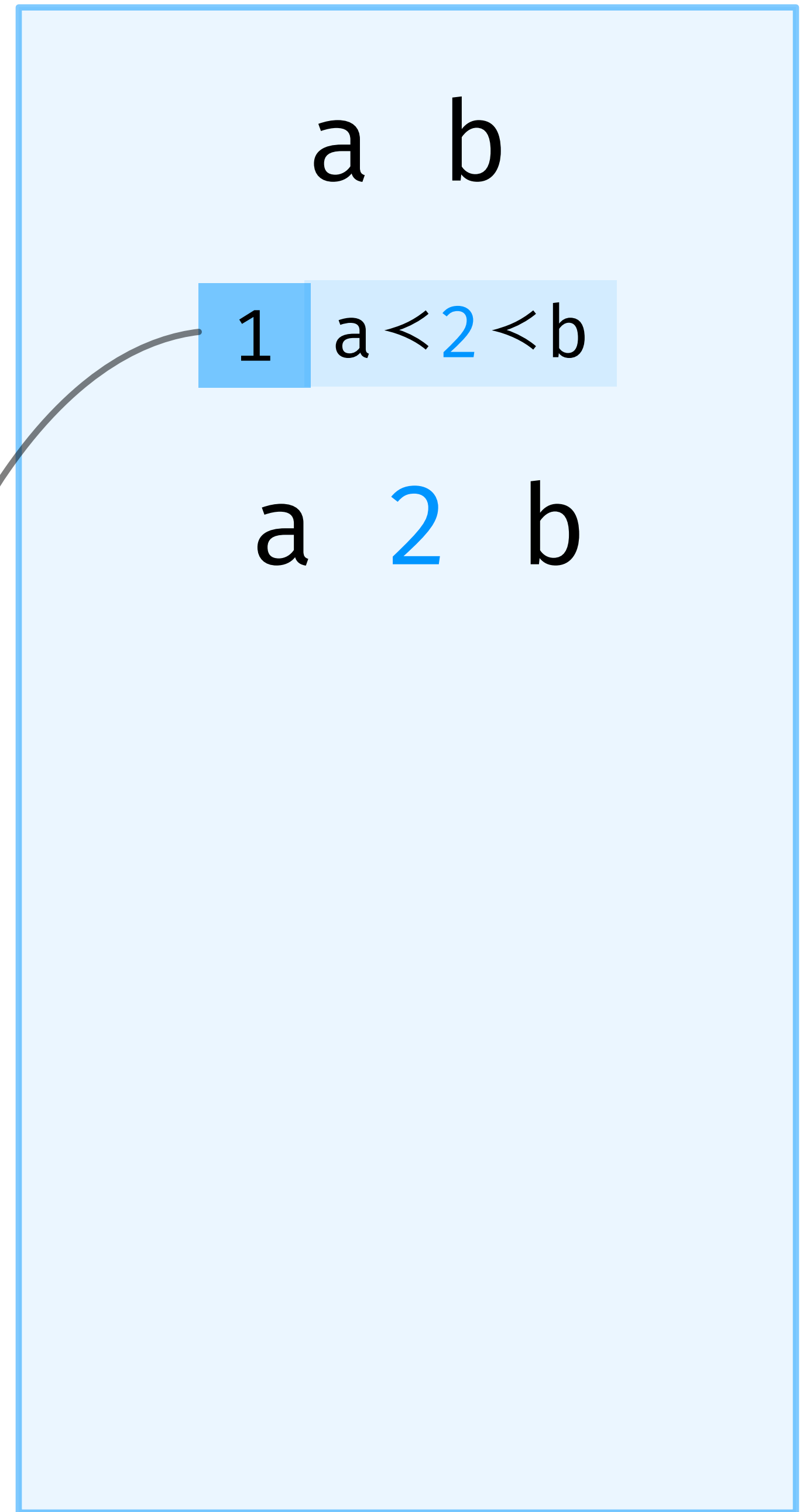
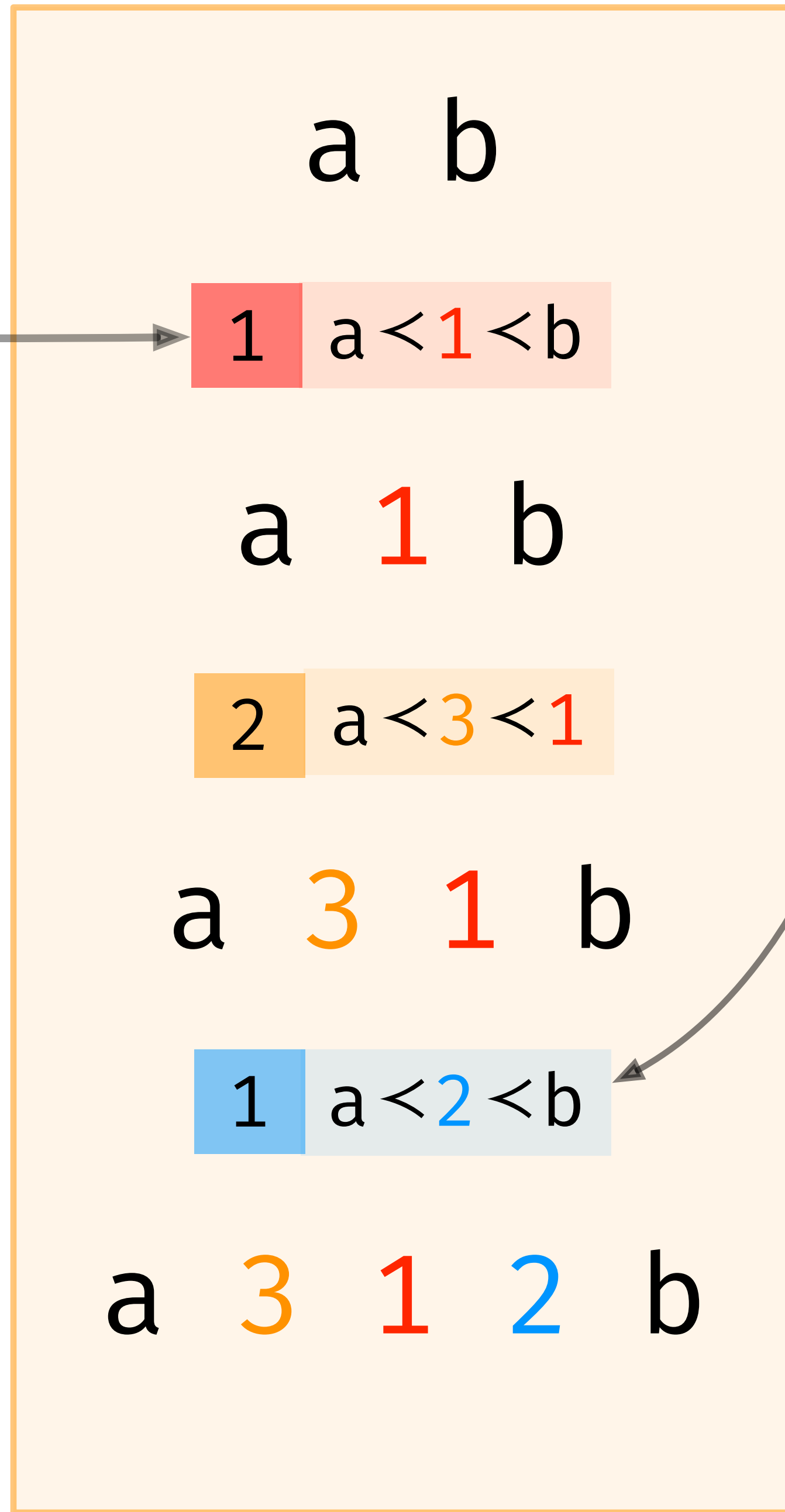
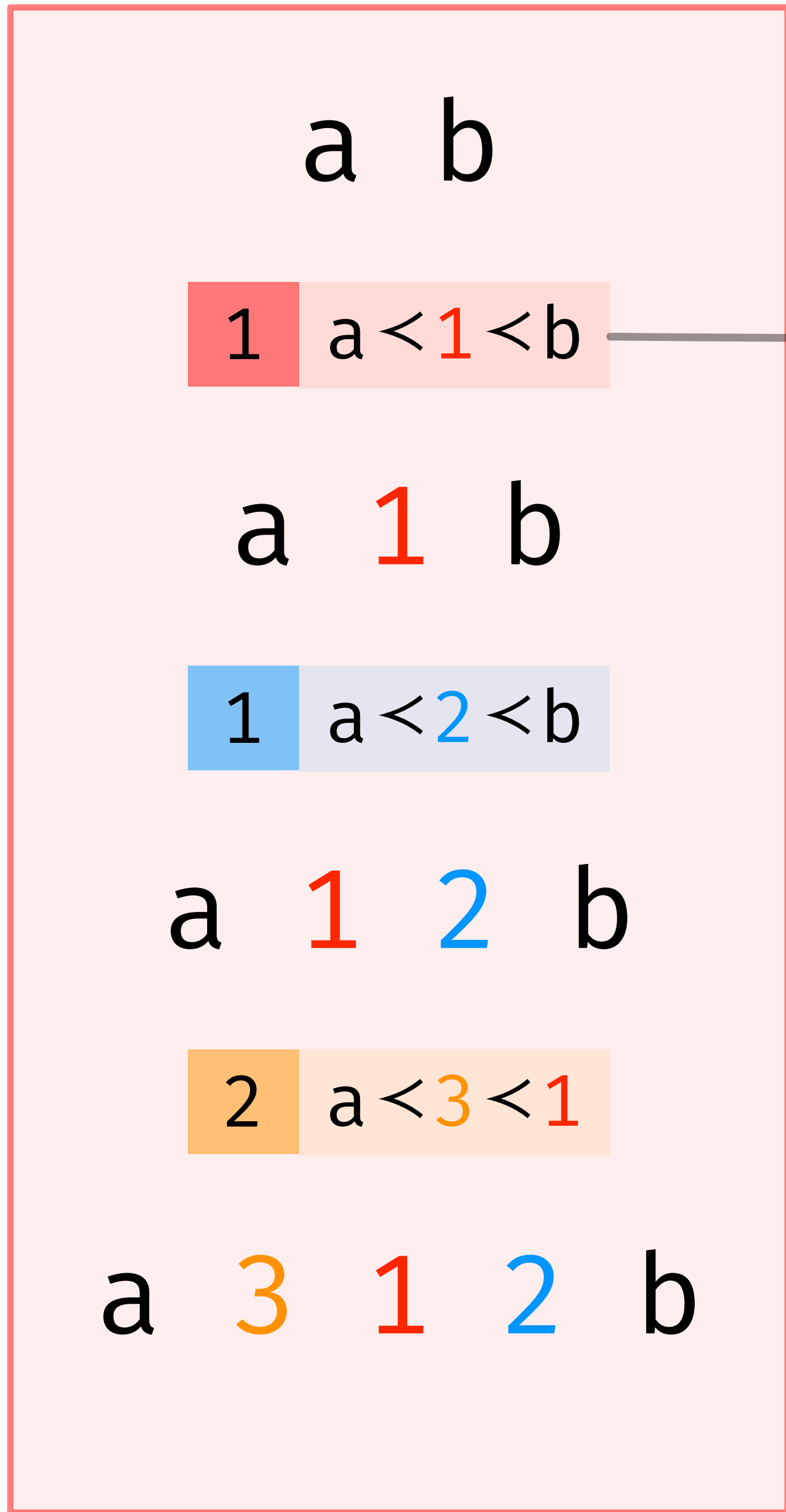












Supporting String-Wise Operations and Selective Undo for Peer-to-Peer Group Editing

Weihai Yu
UiT – The Arctic University of Norway
Weihai.Yu@uit.no

ABSTRACT

Real-time group editing has been envisioned as an effective manner of collaboration. For years, operational transformation (OT) has been the standard concurrency control mechanism for real-time group editing, due to its potential for high responsiveness to local editing operations. OT algorithms are generally non-trivial to be error-free and are computation intensive. Recently, commutative replicated data types (CRDT) have appeared as an alternative to OT. The state-of-the-art OT and CRDT work still lacks the basic functionality found in single-user text editors. In particular, there is no published work that supports both string-wise operations and selective undo. This paper presents an approach that combines and extends OT and CRDT strengths. It is fully decentralized and supports string-wise editing operations and selective undo. Our performance study shows that it provides sufficient responsiveness to the end-users.

Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed Systems—*Distributed applications*; H.5.3 [Information Interfaces and Presentation]: Group and Organization Interfaces—*Collaborative computing*

General Terms

Algorithms, Performance

Keywords

Real-time collaborative editor, commutative replicated data type, operation transformation.

1. INTRODUCTION

A real-time group editor allows multiple users to simultaneously edit the same document from different places. Fully decentralized, or peer-to-peer, collaboration has generally the advantage of availability, scalability and resistance to censorship and surveillance, over collaboration via a central server.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GROUP'14, November 9–12, 2014, Sanibel Island, Florida, USA.
Copyright 2014 ACM 978-1-4503-3043-5/14/11 ...\$15.00.
<http://dx.doi.org/10.1145/2660398.2660401>.

Operational transformation (OT) has been established as a concurrency control mechanism for real-time group editing due to its potential for high responsiveness to local operations [4, 5, 6, 7, 8, 10, 14, 15, 16, 17, 18, 19, 21]. Local operations are executed immediately at local peers and later transformed and integrated at remote peers. OT algorithms are sophisticated. Counterexamples of several published OT algorithms have been reported. Moreover, they have time complexity in the length of operation history of the document been edited, which potentially grows endlessly.

Recently, a new class of mechanisms called commutative replicated data types (CRDT) have been proposed [2, 11, 12, 13, 22, 23]. Concurrent operations of a CRDT are mutually commutative, so that a document is eventually kept consistent at all peers.

A real-time group editor should support at least the most basic functionality found in a single-user text editor. At its minimum, it should support insertion and deletion of single characters and strings of characters, as well as the undo and redo of the insertions and deletions. String-wise operations are important as they are the basis for other useful operations like copy-paste, select-delete and find-replace. Surprisingly, there is currently no published work that supports both string-wise operations and their undo.

Our work supports both string-wise operations and their undo by combining and extending existing OT and CRDT approaches.

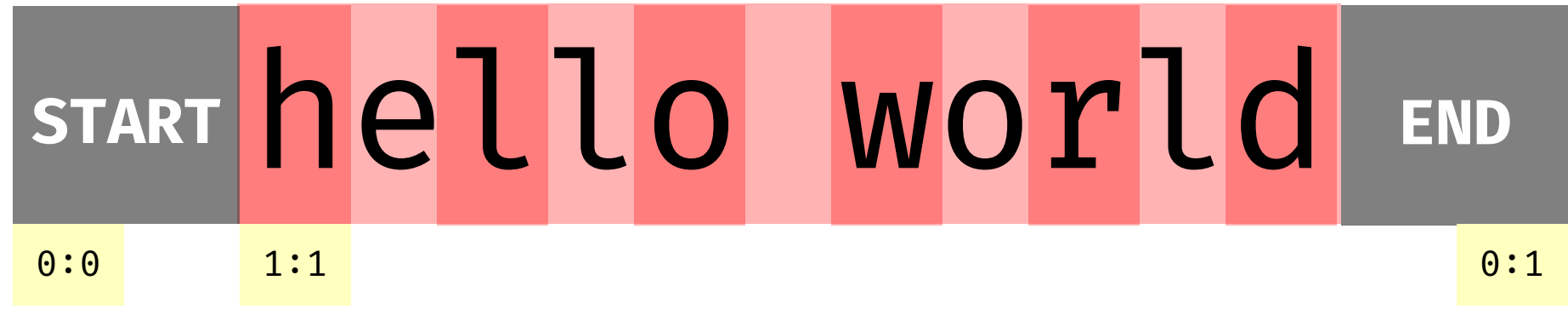
The rest of this paper is organized as follows. Section 2 presents background and related work. Section 3 gives an overview of the approach. Section 4 presents the view-model architecture and the data structure of the model. Section 5 describes operations and updates in view and model. Section 6 describes how model and view are synchronized. Sections 7 and 8 describe how local and remote operations are integrated into the model. Section 9 shows the correctness of the approach. Section 10 presents performance results. Section 11 discusses some open issues. Section 12 concludes.

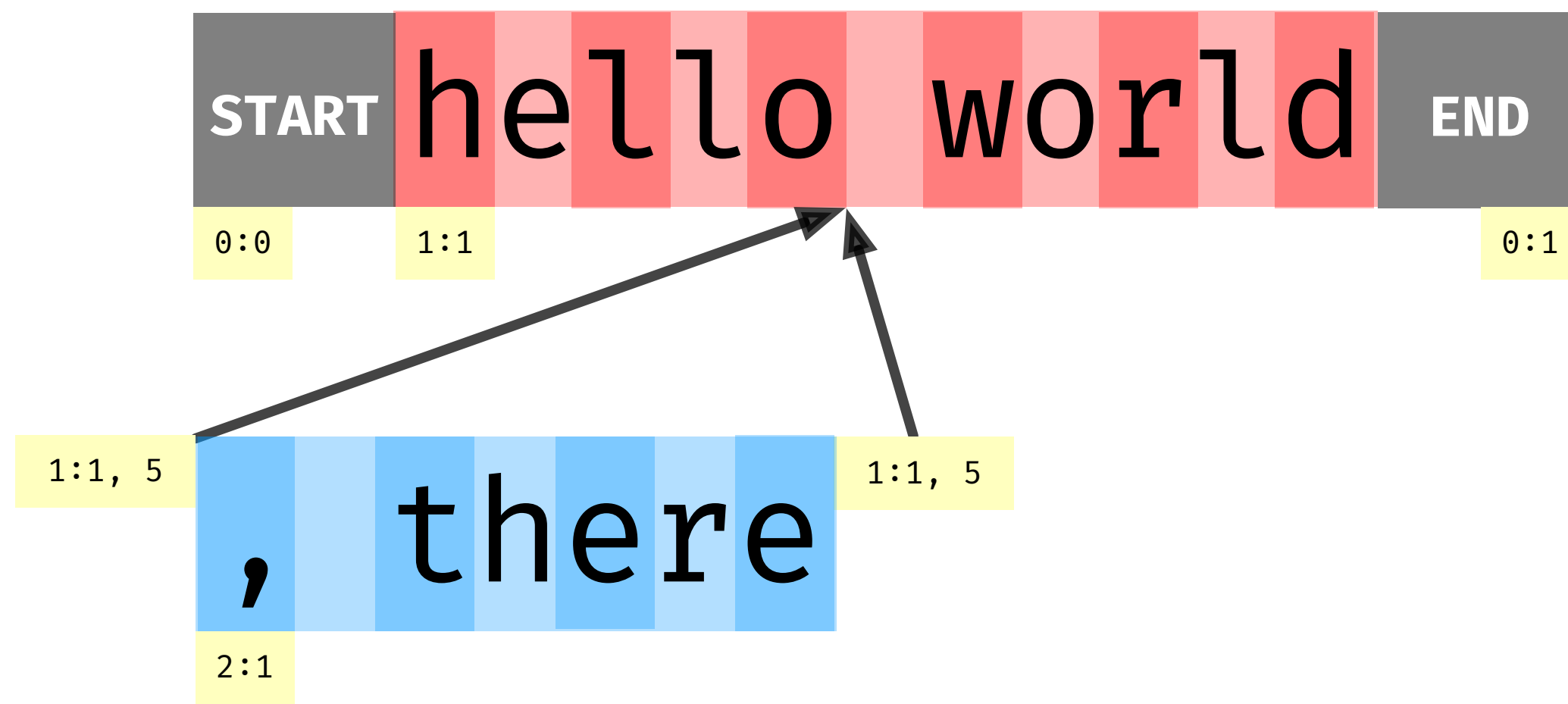
2. BACKGROUND

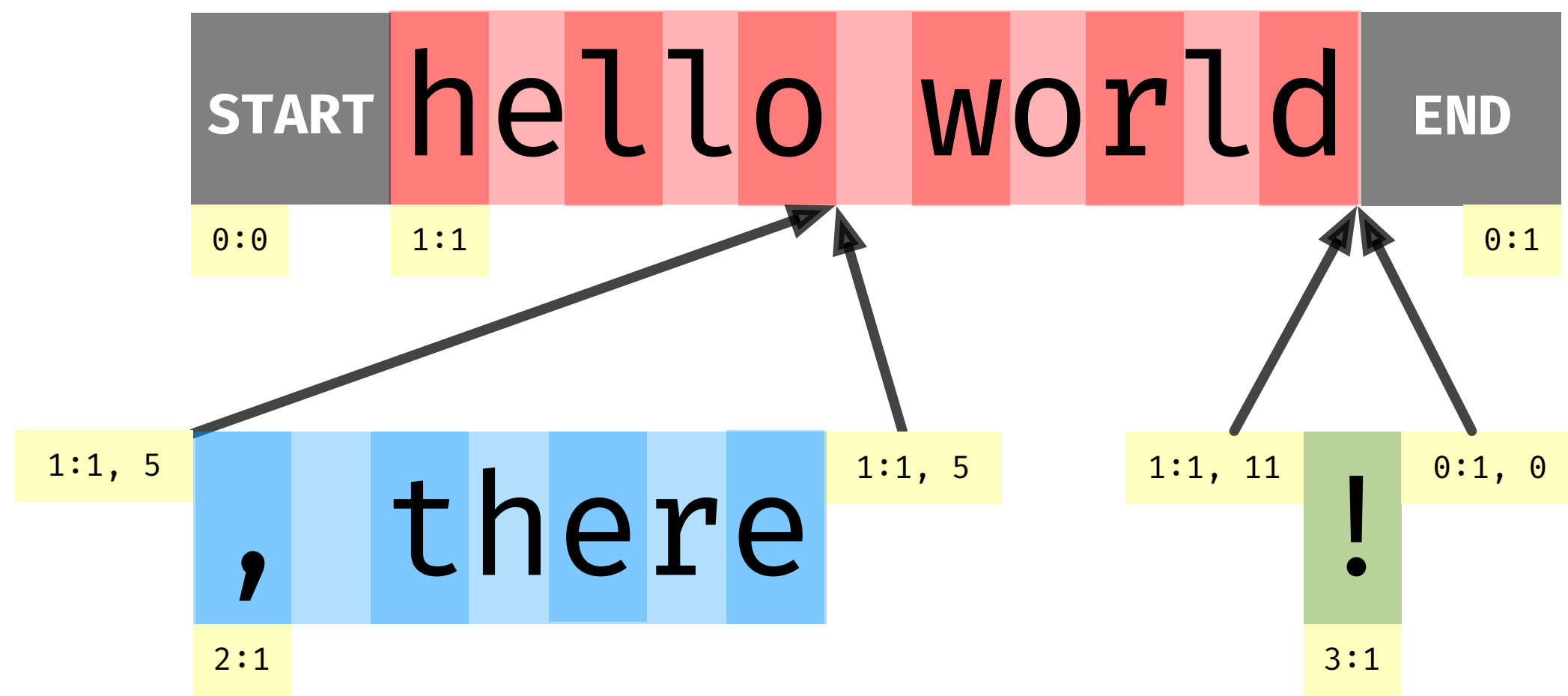
OT was first introduced in [4]. The basic idea is as the following. A shared document is replicated at different peers. An editing operation is first executed at a local peer and then propagated to remote peers. Suppose two peers start with “012”. Peer 1 inserts “a” between “0” and “1” with *ins*(1, “a”) and Peer 2 deletes “2” with *del*(2). The states after local executions at the two peers are “0a12” and “01”. Now if the two peers execute the remote operations as is, the states at these peers become “0a2” and “0a1”, which are inconsistent. With OT, the remote operations are transformed to *include* the executed concurrent operations, into *ins*(1, “a”) and *del*(3) respectively. The two peers are in consistent state “0a1” after executing the transformed operations.

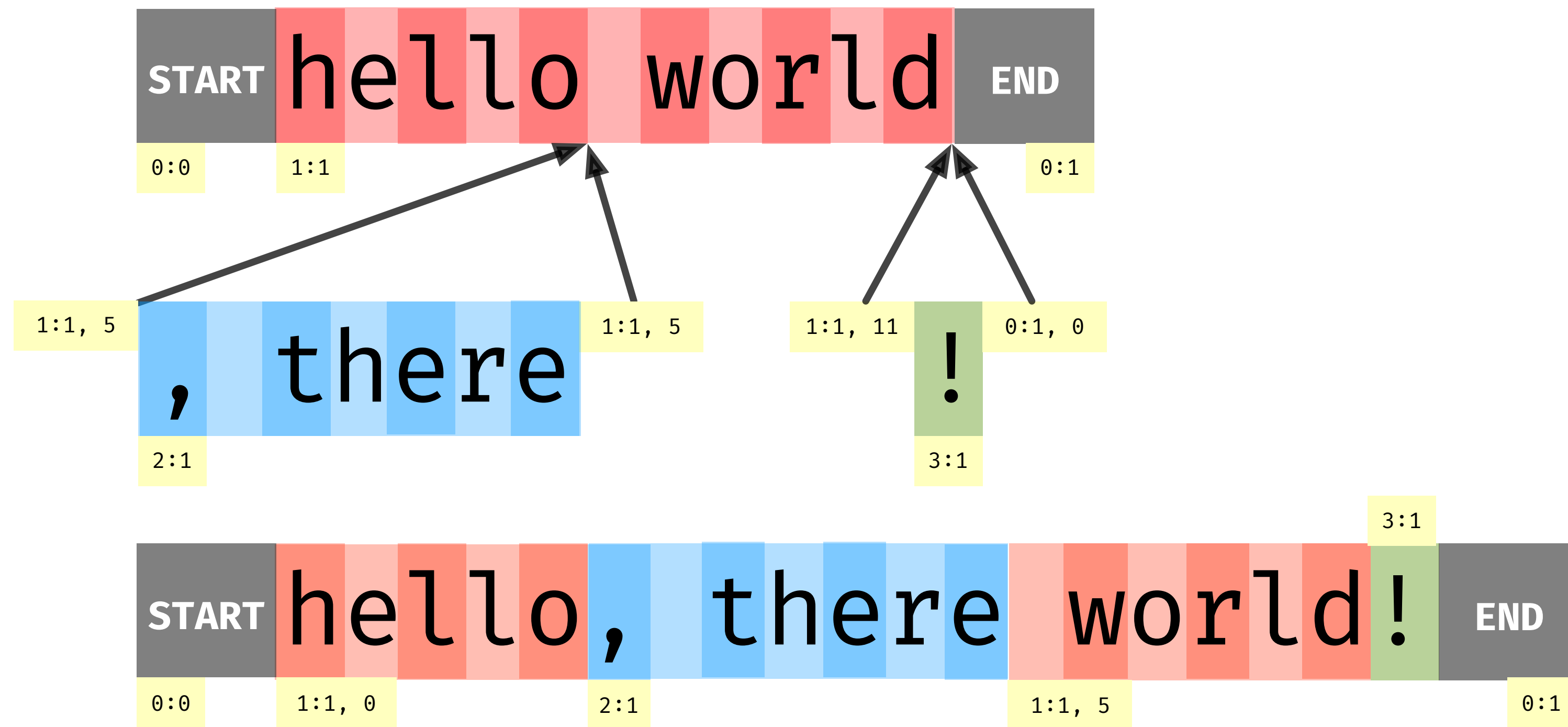
There are some challenges with this basic approach. First, a remote operation can only be transformed to include a concurrent

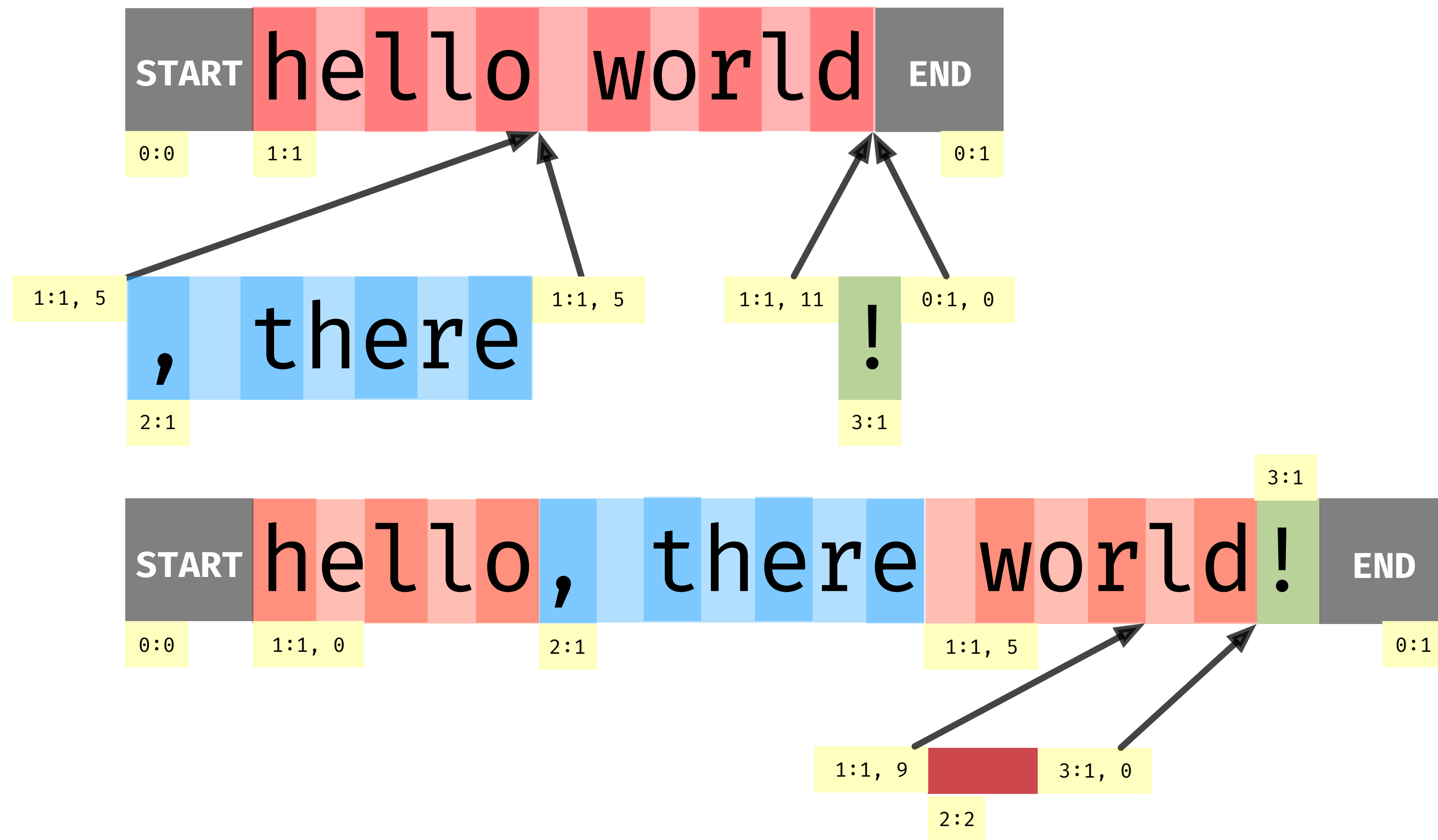
String-Wise WOOT 2014

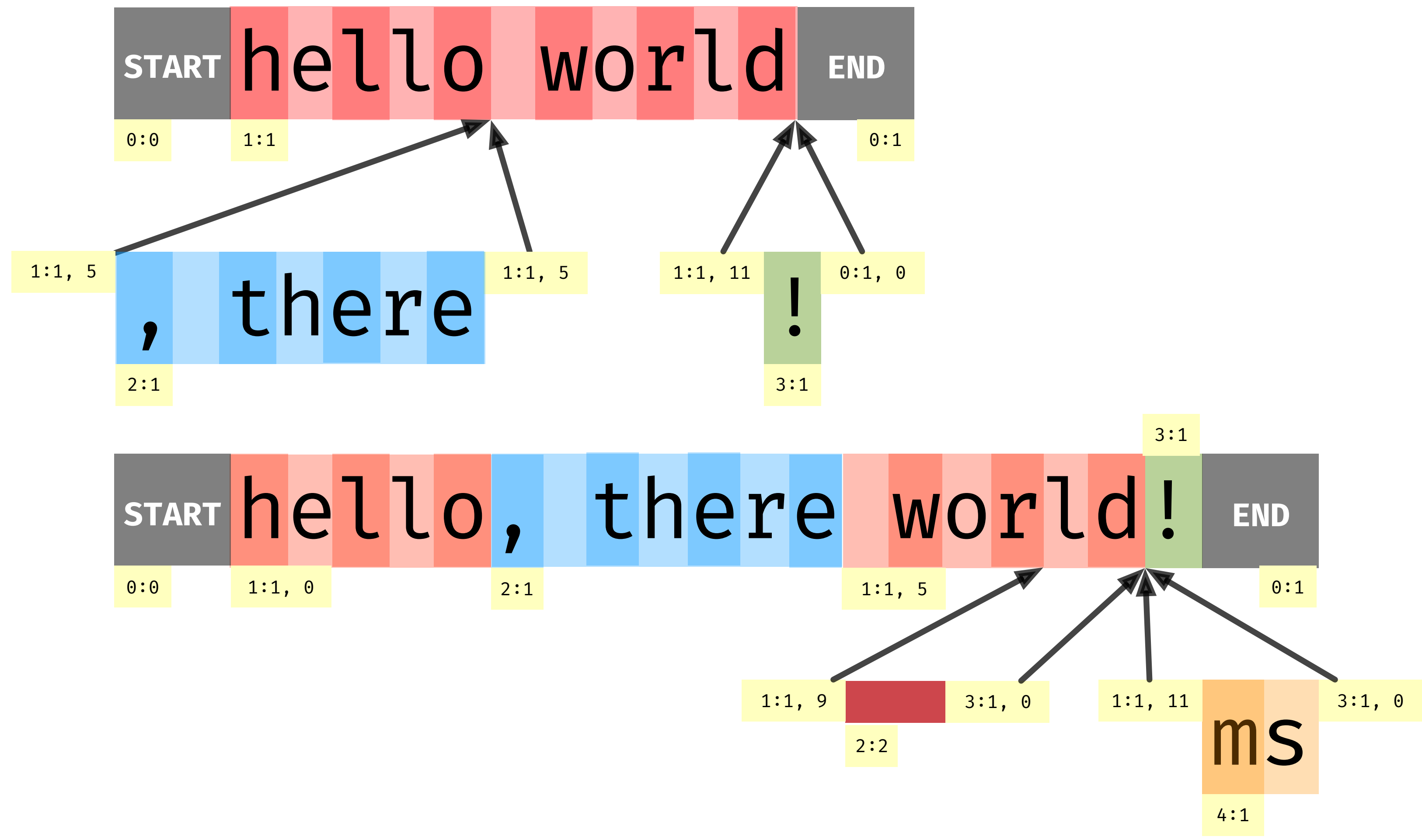


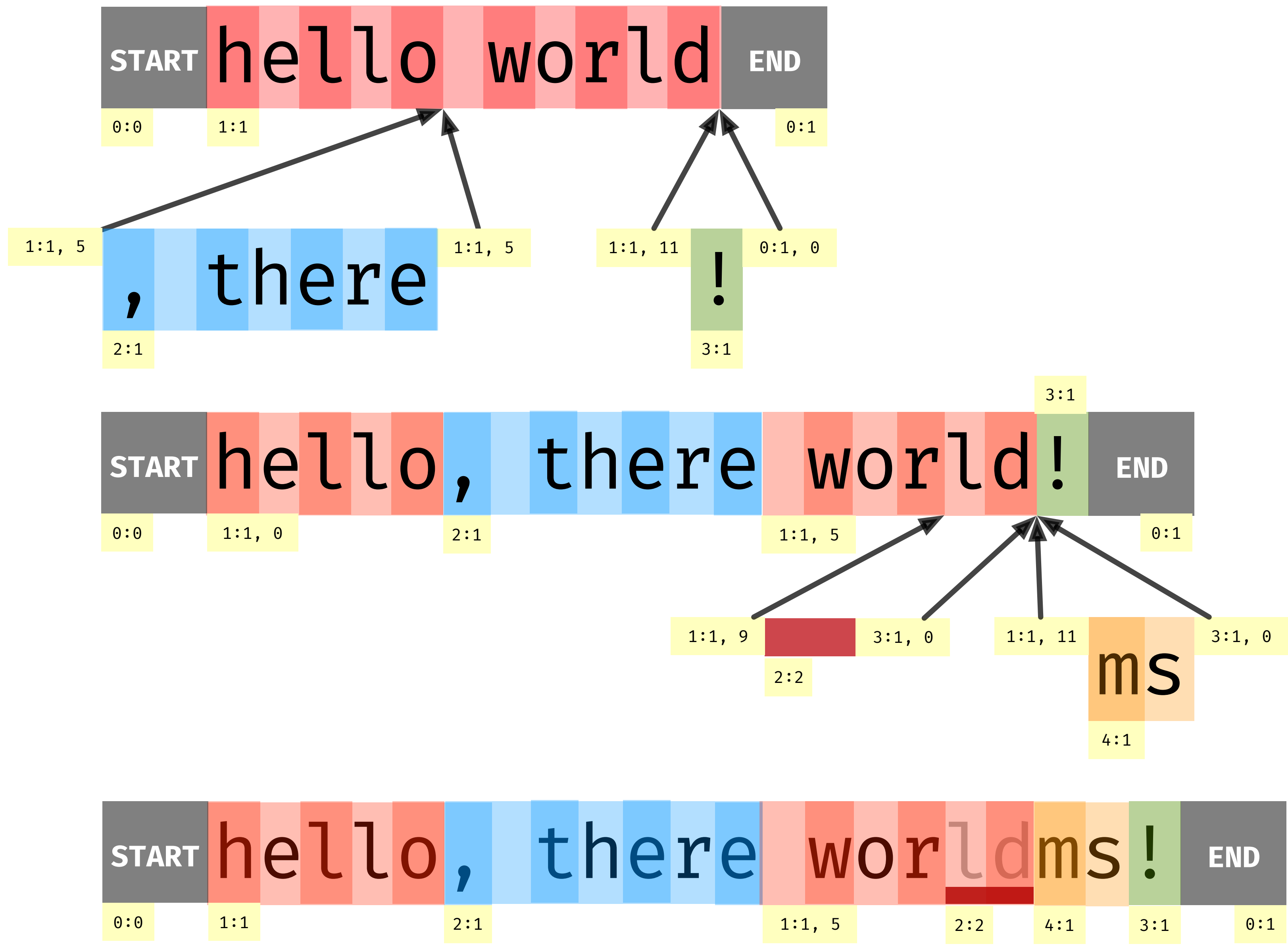






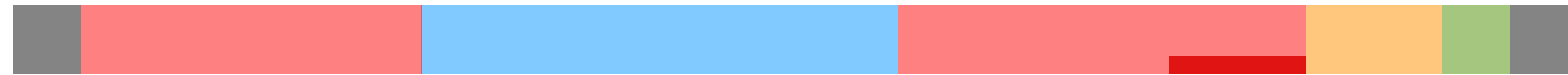


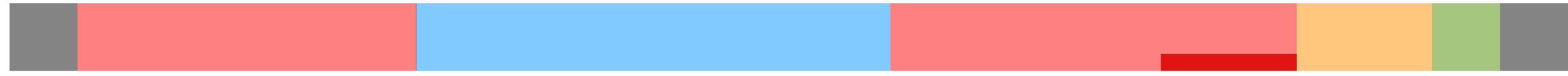


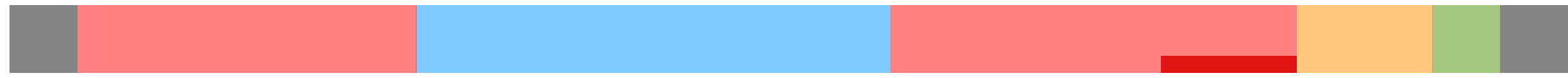


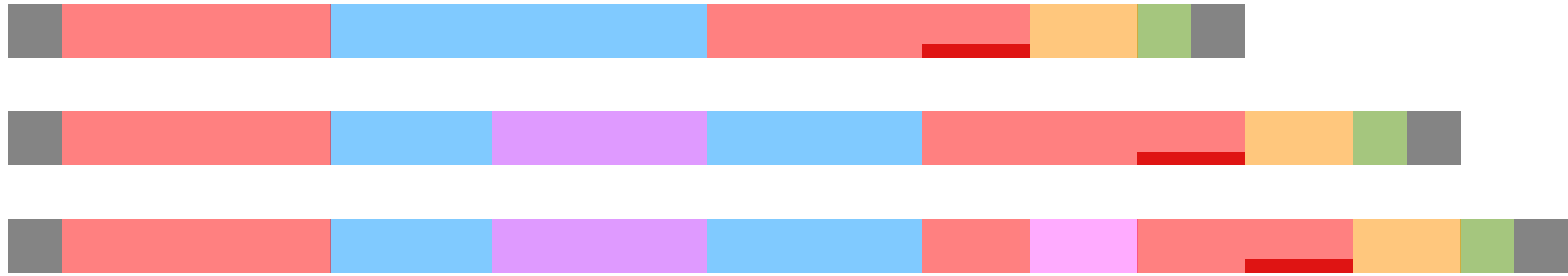
START hello, there worlds! END

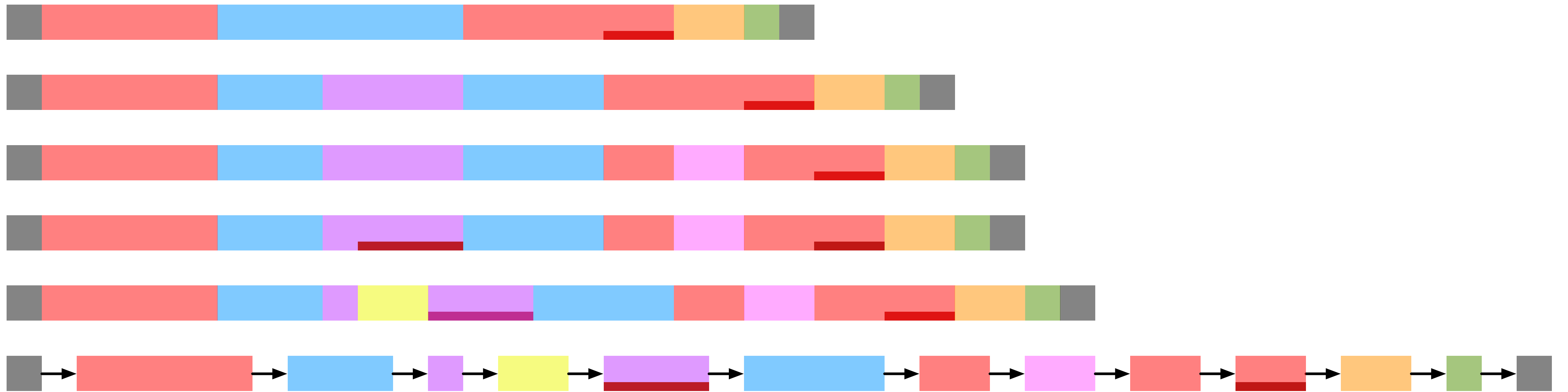
START hello, there worlds! END

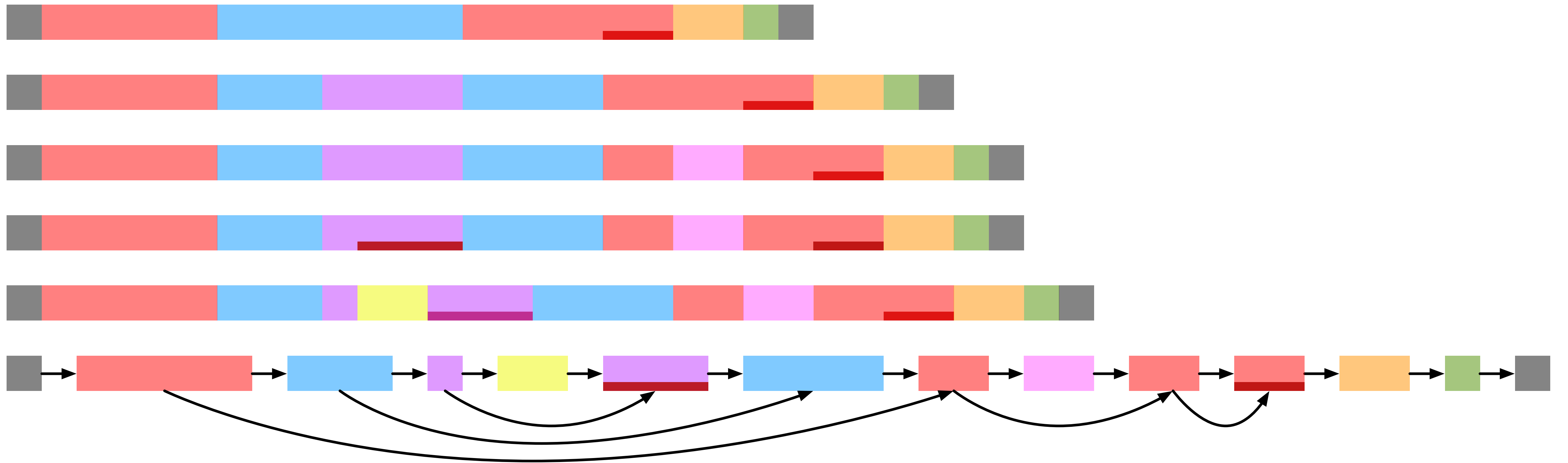


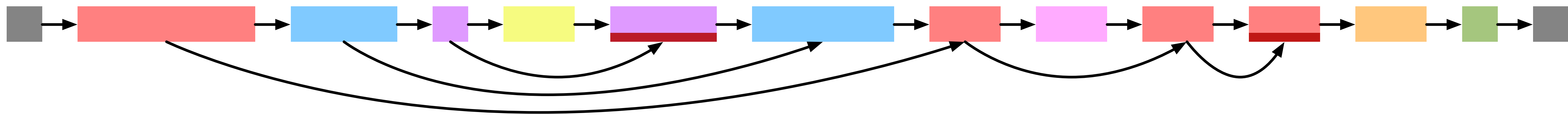


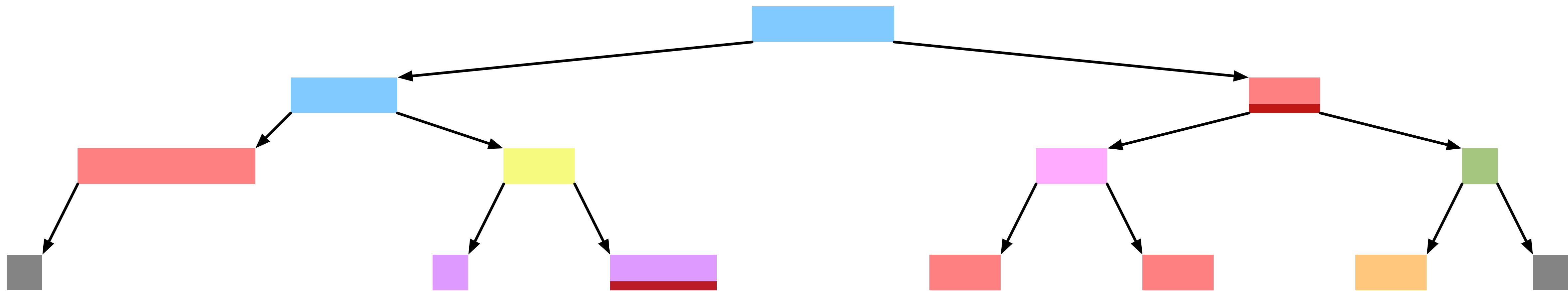
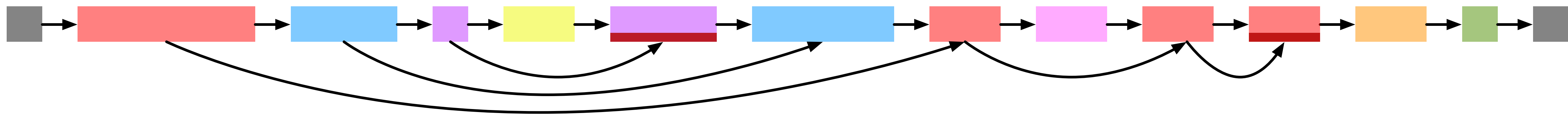


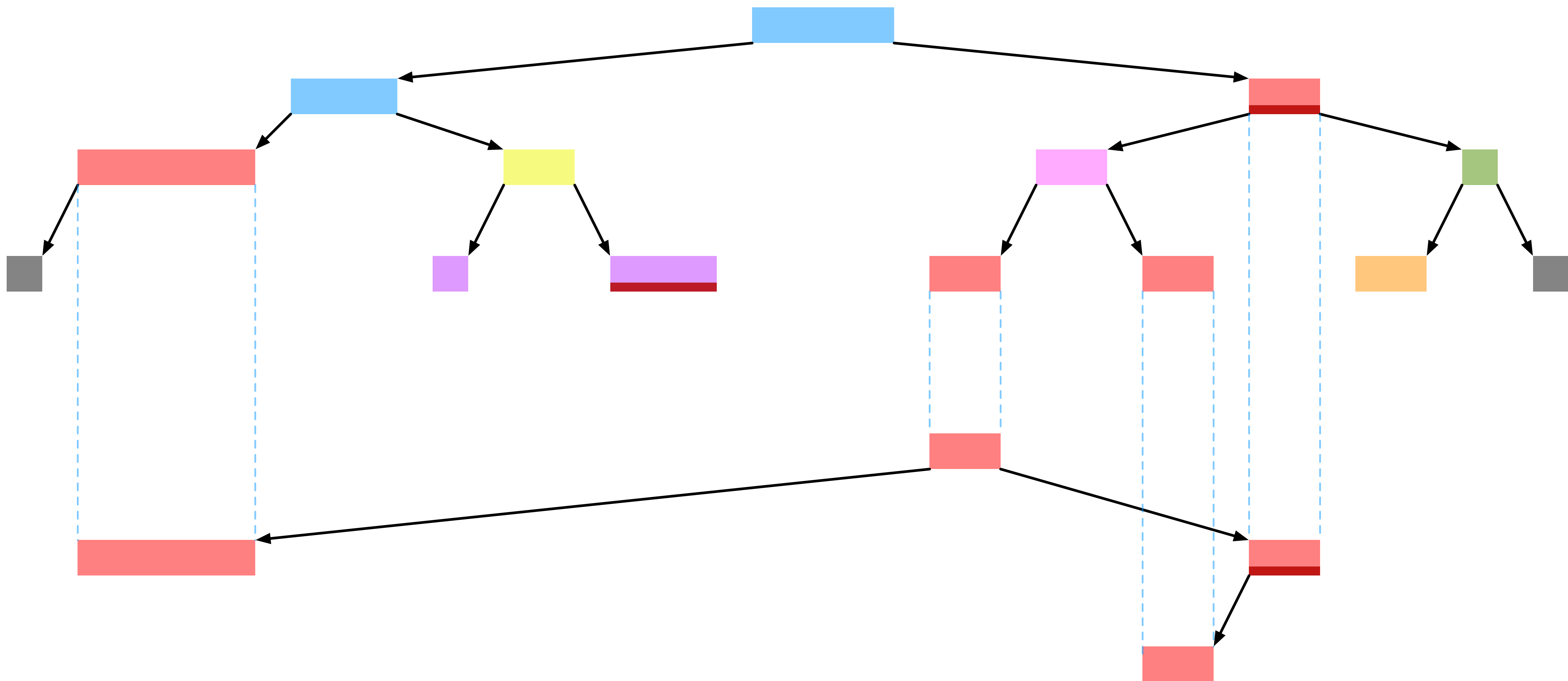
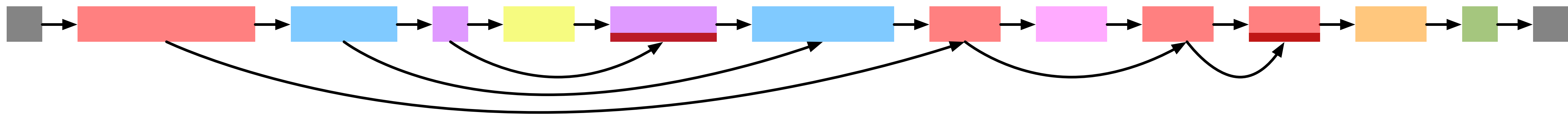


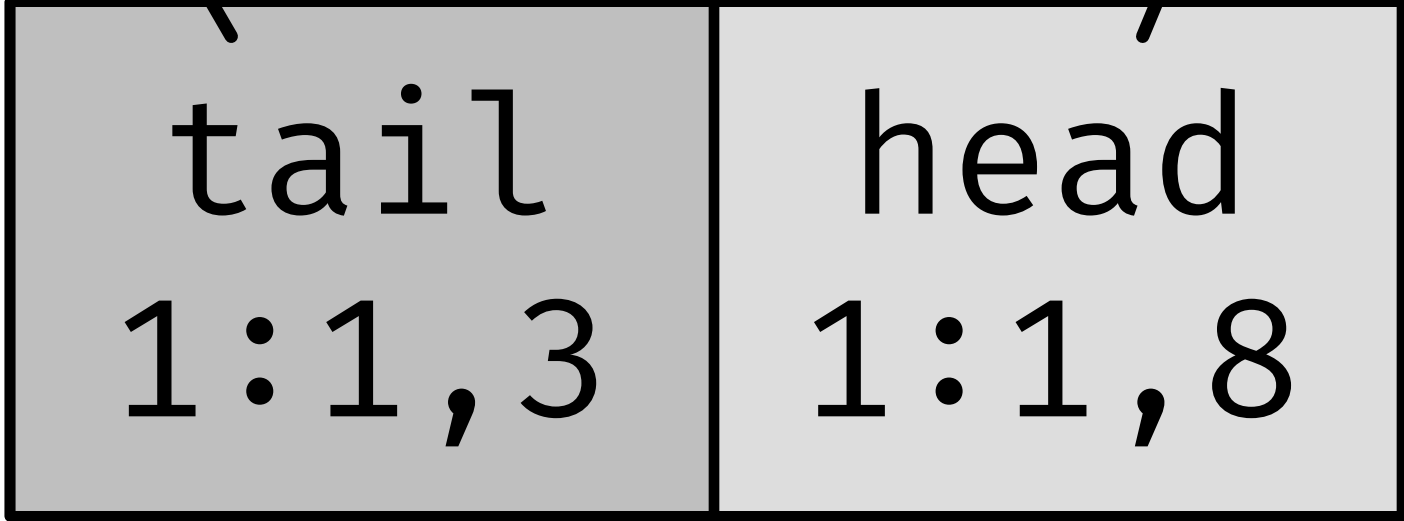
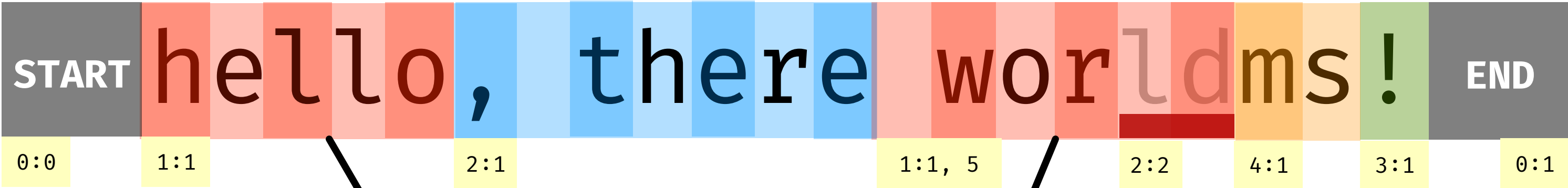










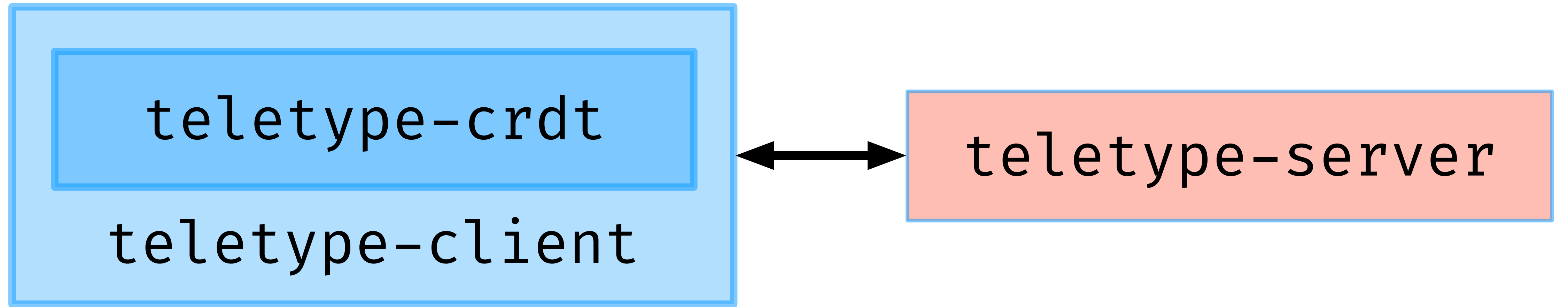


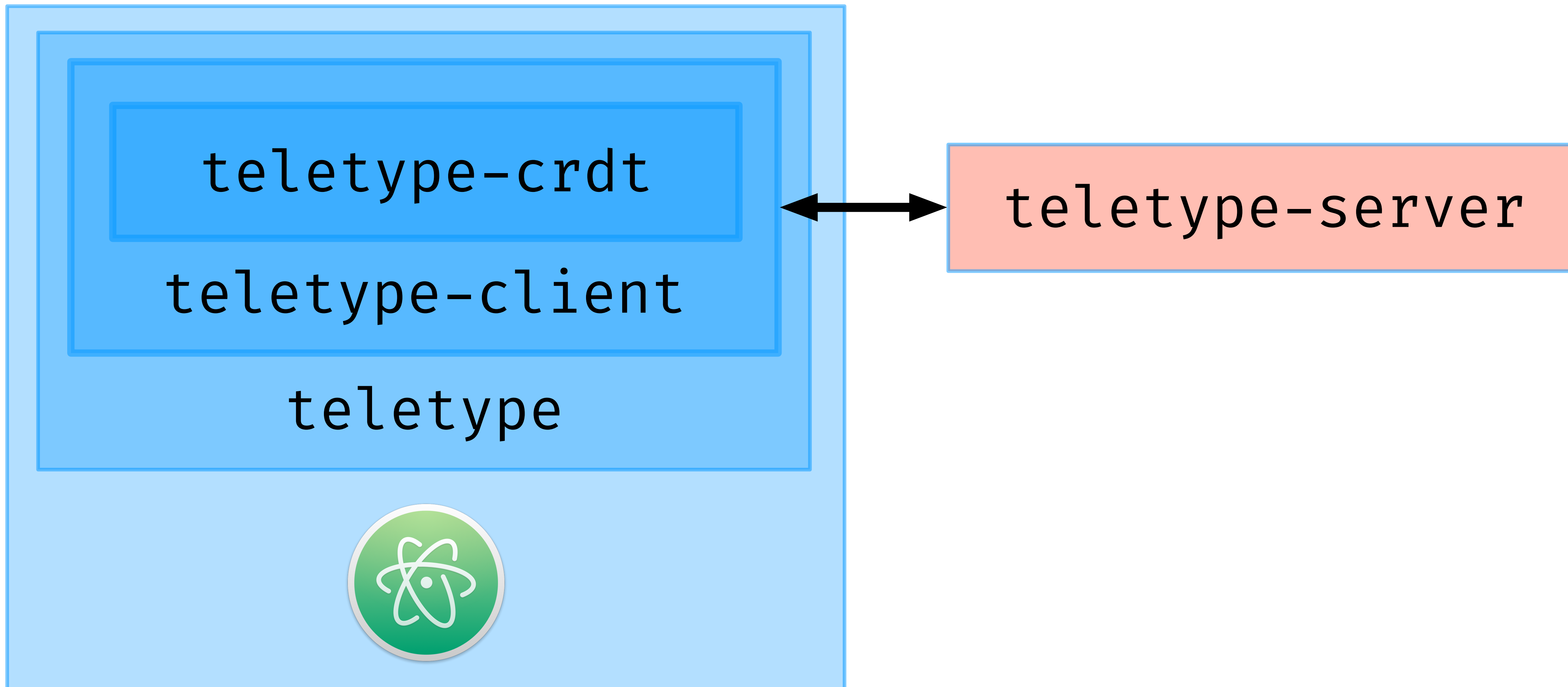


teletype-crdt

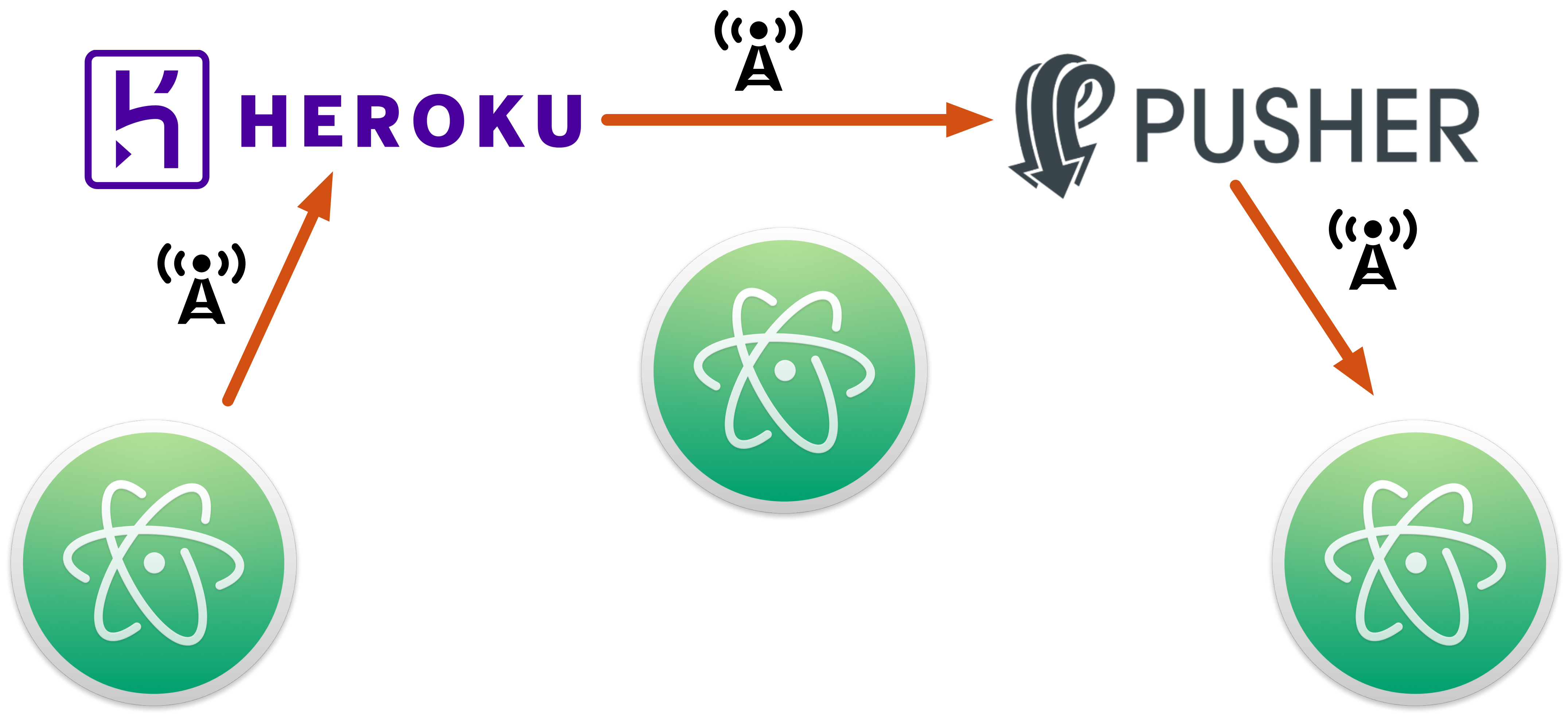
teletype-crdt

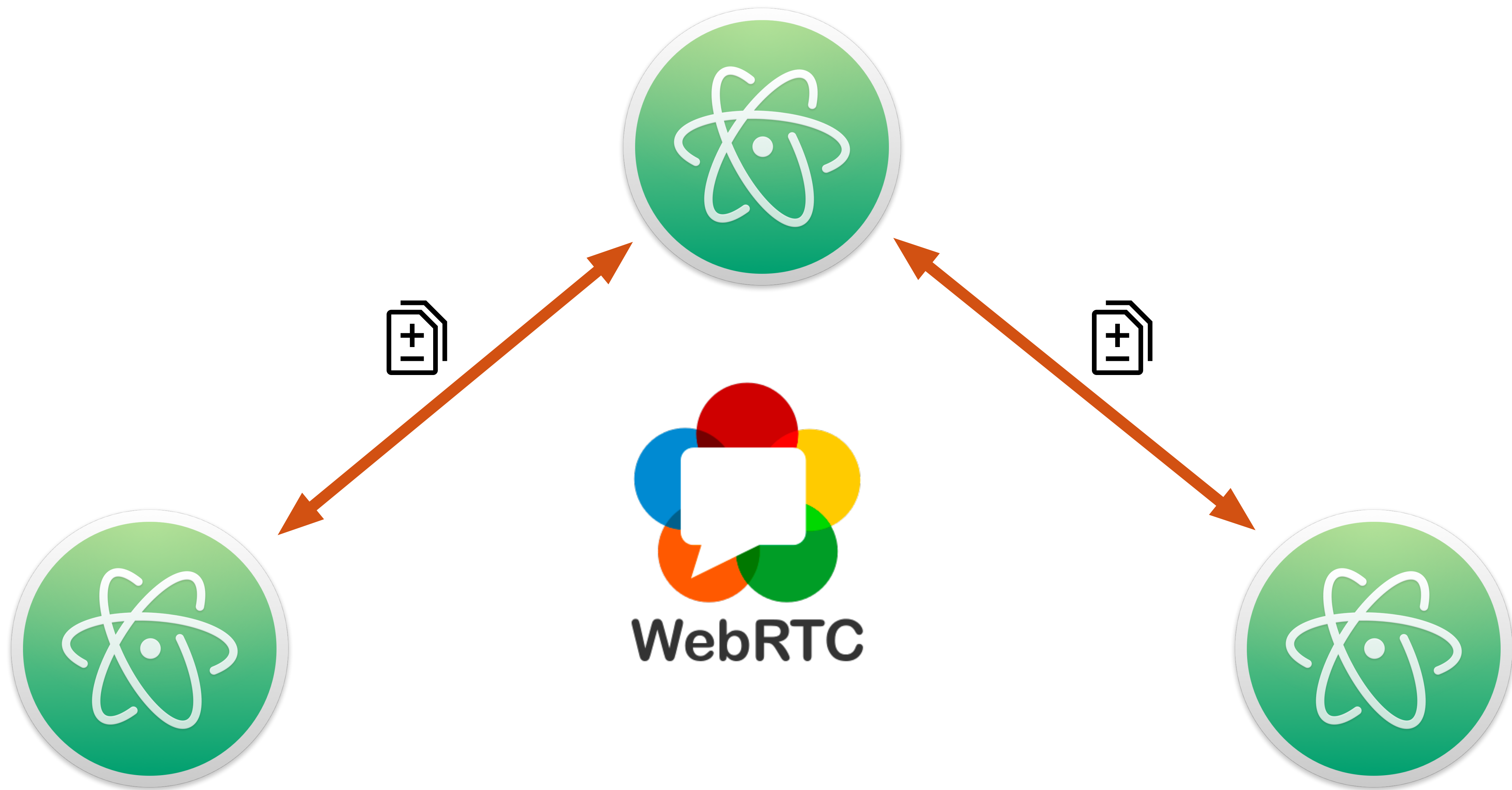
teletype-client











SPRAY: an Adaptive Random Peer Sampling Protocol

Brice Nédelec
Université de Nantes, LINA
brice.nedelec@univ-nantes.fr

Julian Tanke
Université de Nantes, LINA
julian.tanke@fu-berlin.de

Davide Frey
INRIA Bretagne-Atlantique
davide.frey@inria.fr

Pascal Molli
Université de Nantes, LINA
pascal.molli@univ-nantes.fr

Achour Mostefaoui
Université de Nantes, LINA
achour.mostefaoui@univ-nantes.fr

ABSTRACT

The introduction of WebRTC has opened a new playground for large-scale distributed applications consisting of large numbers of directly-communicating web browsers. In this context, gossip-based peer-sampling protocols appear as a particularly promising tool thanks to their inherent ability to build overlay networks that can cope with network dynamics. However, the dynamic nature of browser-to-browser communication combined with the connection establishment procedures that characterize WebRTC make current peer-sampling solutions inefficient or simply unreliable. In this paper, we address the limitations of current peer-sampling approaches by introducing SPRAY, a novel peer-sampling protocol designed to avoid the constraints introduced by WebRTC. Unlike most recent peer-sampling approaches, SPRAY has the ability to adapt its operation to networks that can grow or shrink very rapidly. Moreover, by using only neighbor-to-neighbor interactions, it limits the impact of the three-way connection establishment process that characterizes WebRTC. Our experiments demonstrate the ability of SPRAY to adapt to dynamic networks and highlight its efficiency improvements with respect to existing protocols.

Categories and Subject Descriptors

C.2.1 [Computer Systems Organization]: Computer-Communication Networks—*Network Architecture and Design*

General Terms

Network, algorithm, simulation

Keywords

Large scale distributed applications, random peer sampling, browser-to-browser communication, WebRTC

1. INTRODUCTION

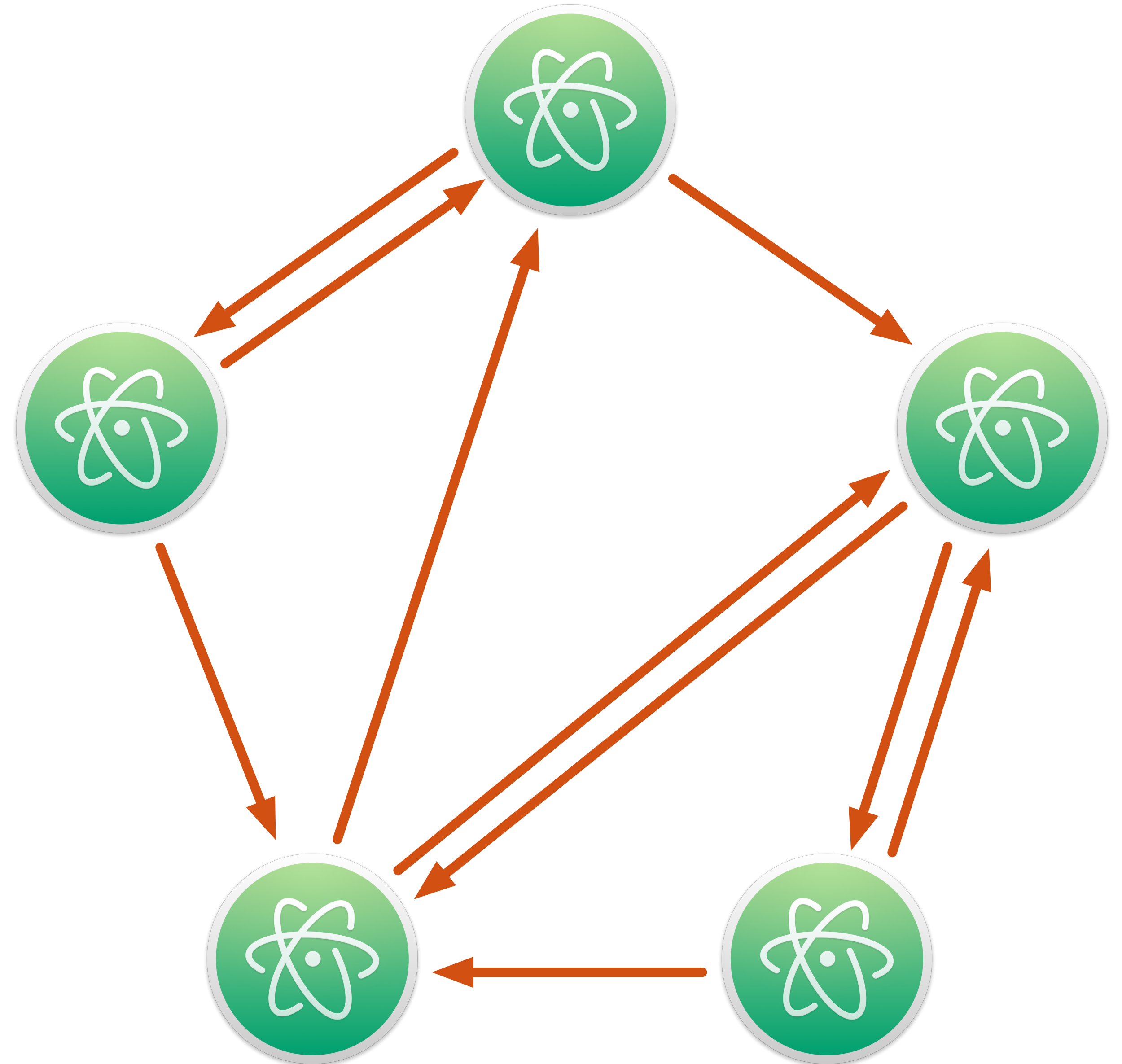
Peer sampling constitutes a fundamental mechanism for many large-scale distributed applications both on the cloud [4] and in a peer-to-peer setting. Services such as dissemination [6, 19], aggregation [12] and network management [13, 21] have been based on peer sampling and the recent introduction of WebRTC¹ opens the opportunity to deploy such applications on browsers that can run on laptops, desktops and mobile devices. In this context, WebRTC drastically simplifies deployment even within complex network systems that utilize firewalls, proxies and Net Address Translation (NAT).

Unfortunately, WebRTC has several constraints that make existing peer sampling services inefficient or unreliable. Browsers can run on small devices in mobile networks. Hence, keeping the number of connections as low as possible is a major requirement. However, peer sampling services such as CYCLON [20] do not adapt the number of connections to the real number of participants. For instance, a user must maintain 10 connections with other remote browsers when only 6 are enough. On the other hand, the peer sampling service SCAMP [10] is adaptive but uses random dissemination paths to establish connections which is much more costly and likely to fail in the WebRTC context.

In this paper, we introduce SPRAY, a random peer sampling protocol inspired by both SCAMP [10] and CYCLON [20]. Compared to the state of art, (i) SPRAY dynamically adapts the neighborhood of each peer. Thus, the number of connections grows logarithmically compared to the size of the network. (ii) SPRAY only uses neighbor-to-neighbor interactions to establish connections. Thus, the connections are established in constant time. (iii) SPRAY quickly converges to a topology exposing properties similar to those of a random graph. Thus, the network becomes robust to massive failures, efficiently disseminates information etc. (iv) In the experimental setup, we show the adaptiveness of SPRAY and highlight its efficiency improvement compared to CYCLON and SCAMP, at the cost of little overhead.

The rest of this paper is organized as follows: Section 2 reviews the related work. Section 3 details the SPRAY protocol. Section 4 shows the properties of SPRAY and compares them to state-of-the-art random peer sampling approaches. We conclude and discuss about the perspective in Section 5.

¹<http://www.webrtc.org/>



A vibrant, mid-century modern style illustration of a futuristic living room. On the left, a large, slanted window shows a man in blue swim trunks and a diving mask swimming in clear blue water. In the center, a tall, leafy green plant stands next to a vintage television set on a wooden stand. To the right, a woman in a bright yellow dress stands near a man in a dark suit who is sitting in a blue armchair, holding a small object. The room features a yellow sofa with a white circular object on it, a round glass coffee table with a blue plate and a magazine, and a textured orange floor. The walls are wood-paneled, and there are floating white rectangular objects. The overall aesthetic is clean, bright, and optimistic.

Future Directions



teletype.atom.io