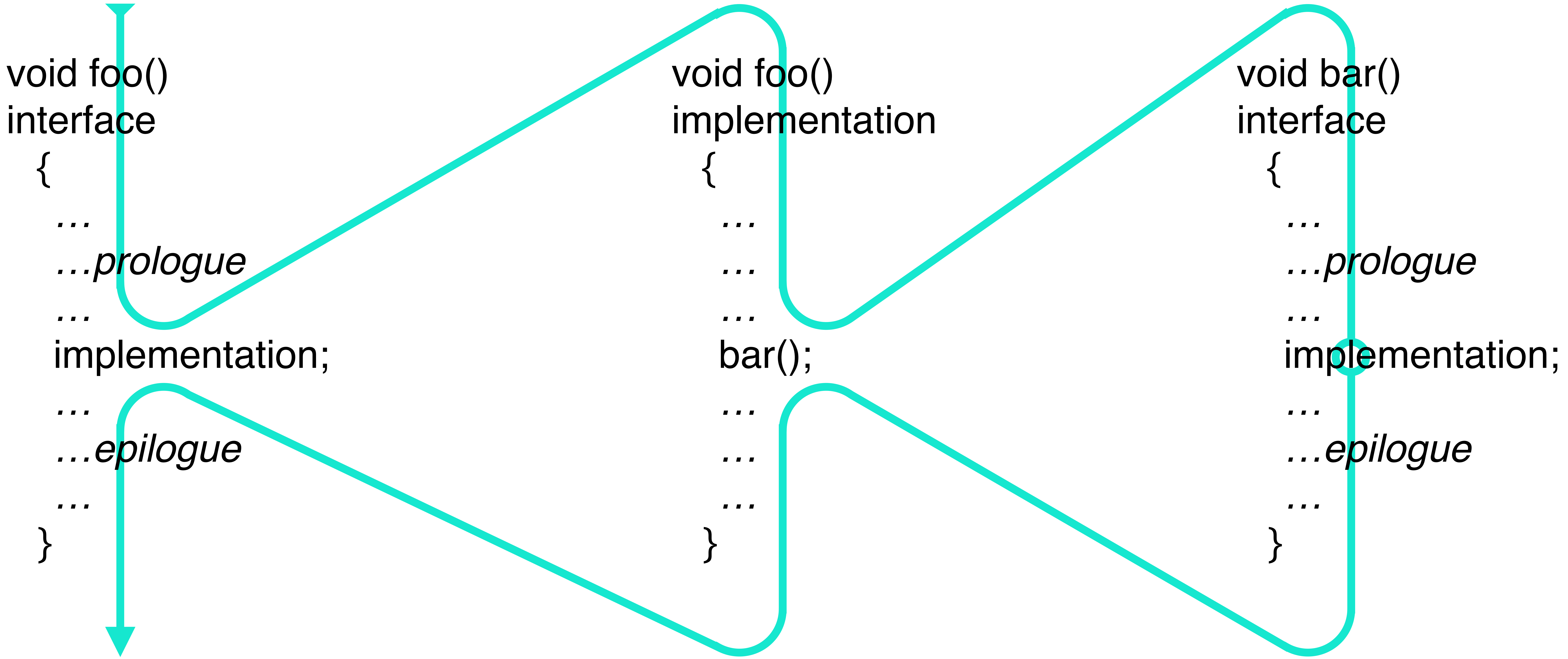
Neighborhoods Banding Together

Reasoning Globally about Programs

Lisa Lippincott

CppCon, September 2020

The code here is written in a fantasy C++,
with extensions supporting local reasoning.

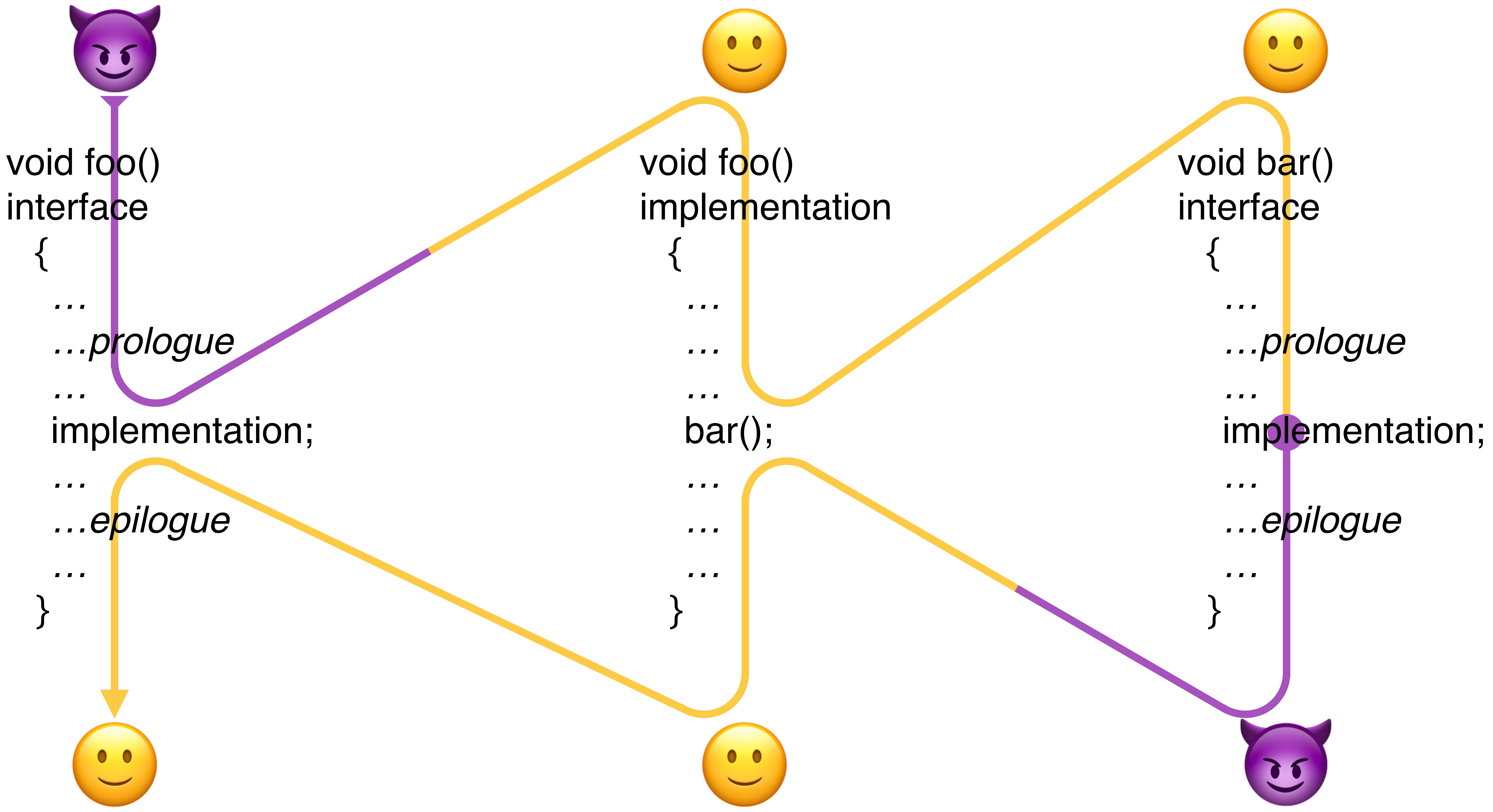


```
void foo()
interface
  {
    …
    …prologue
    …
    implementation;
    …
    …epilogue
    …
  }
```

```
void foo()
implementation
  {
    …
    …
    …
    bar();
    …
    …
    …
  }
```

```
void bar()
interface
  {
    …
    …prologue
    …
    implementation;
    …
    …epilogue
    …
  }
```

See “The Truth of a Procedure”



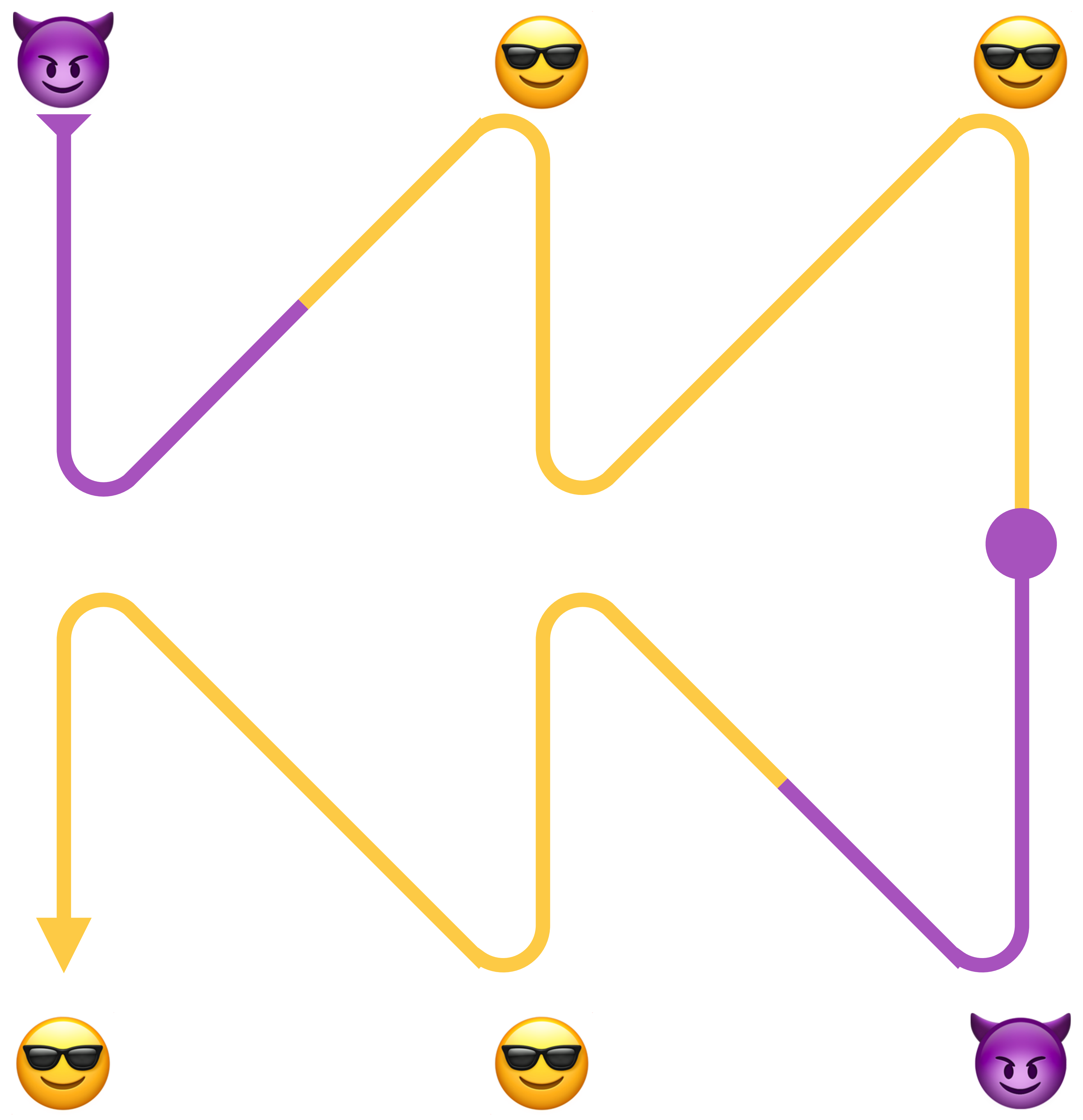
```
void foo()
interface
  {
    …
    …prologue
    …
    implementation;
    …
    …epilogue
    …
  }
```

```
void foo()
implementation
  {
    …
    …
    …
    bar();
    …
    …
    …
  }
```

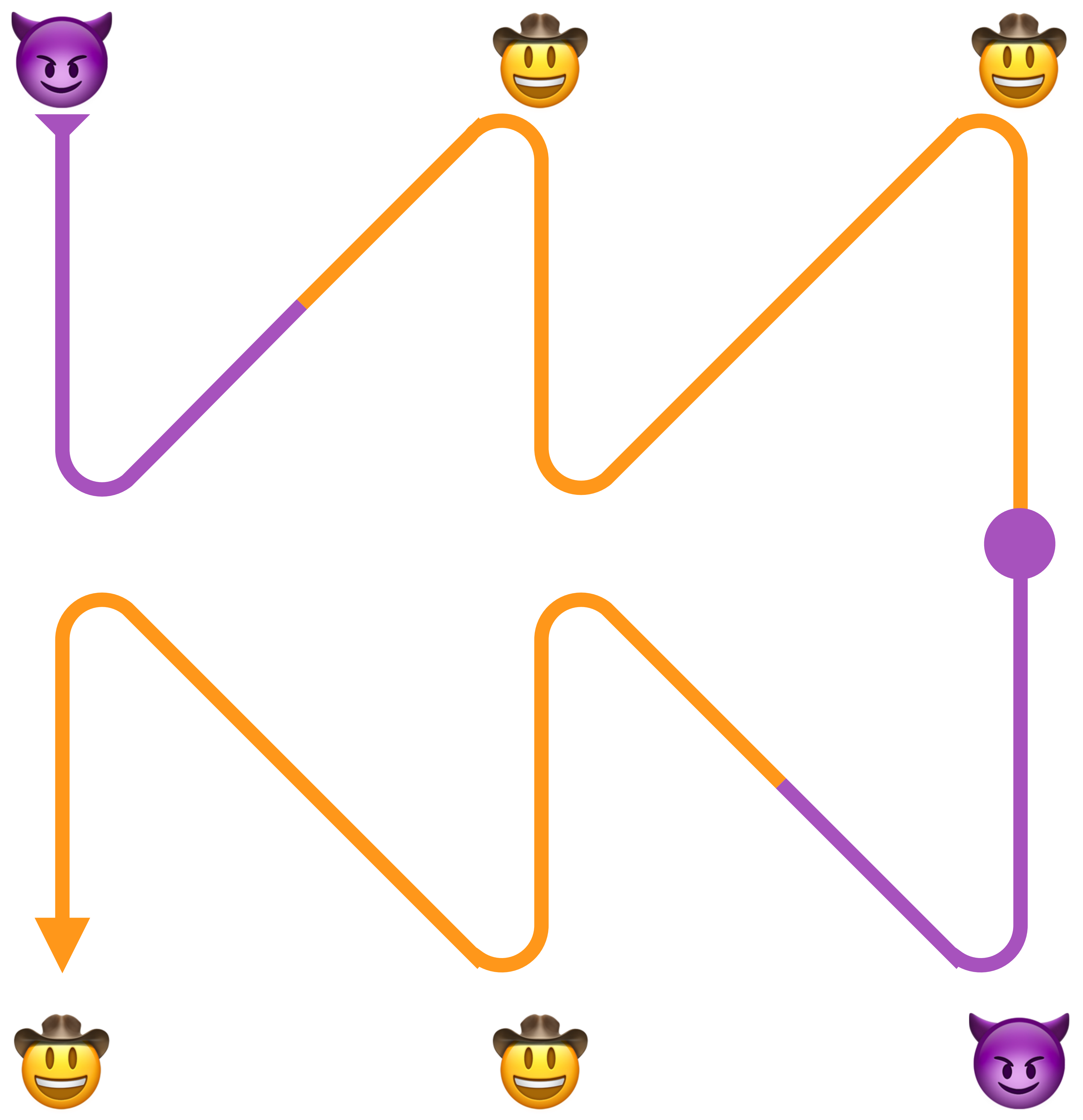
```
void bar()
interface
  {
    …
    …prologue
    …
    implementation;
    …
    …epilogue
    …
  }
```

See “The Truth of a Procedure”

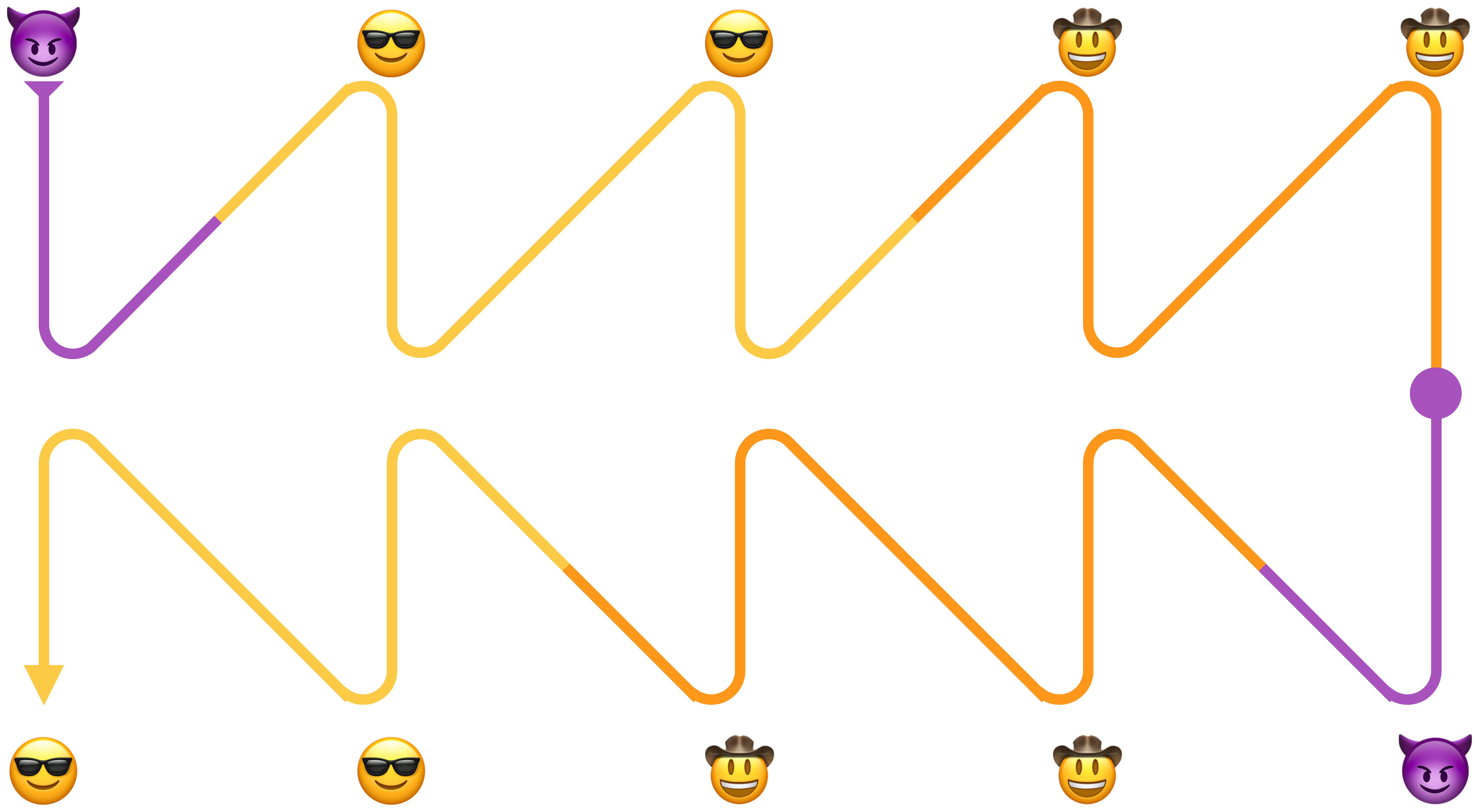
😎 Caller



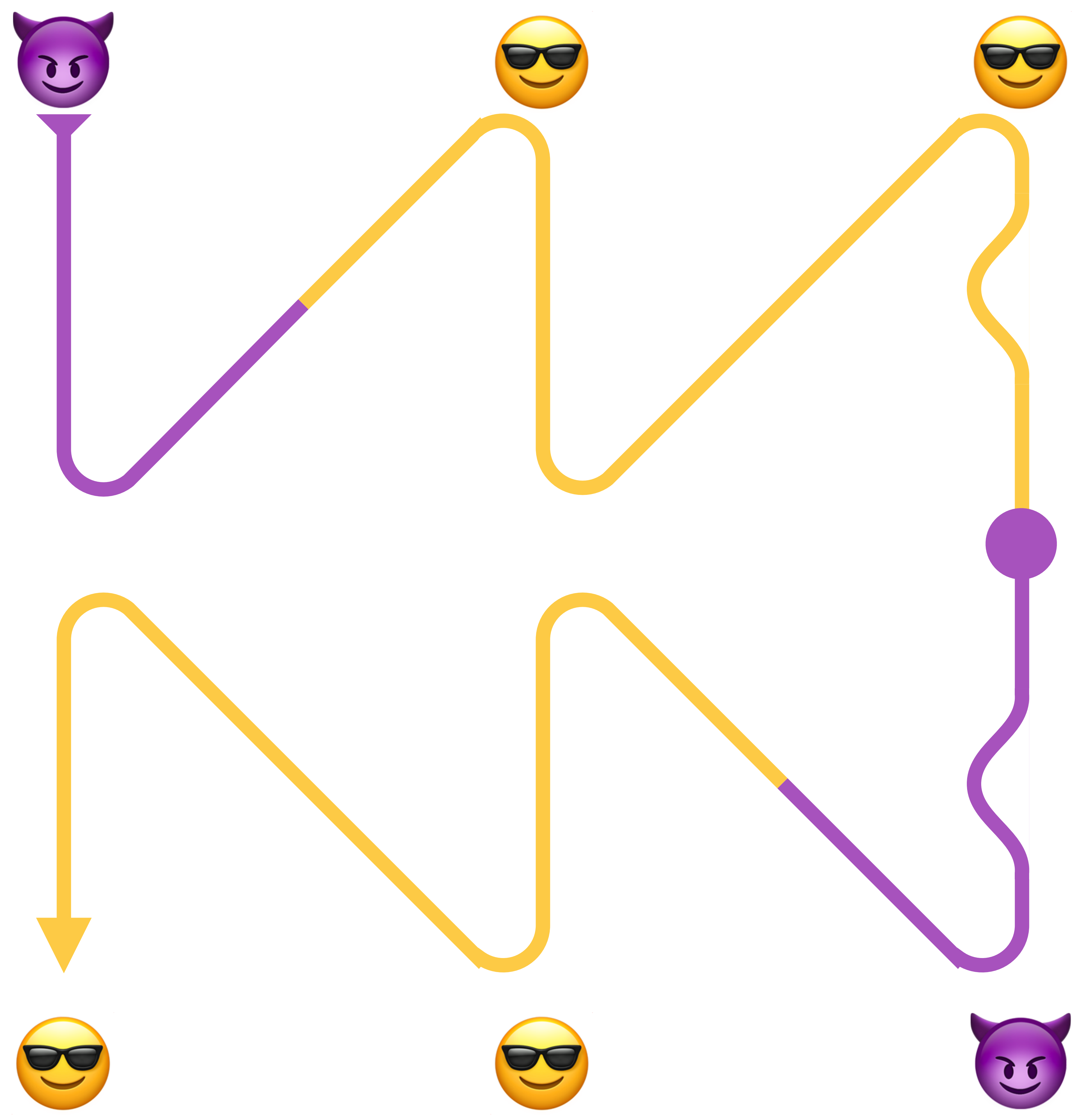
🤠 Called



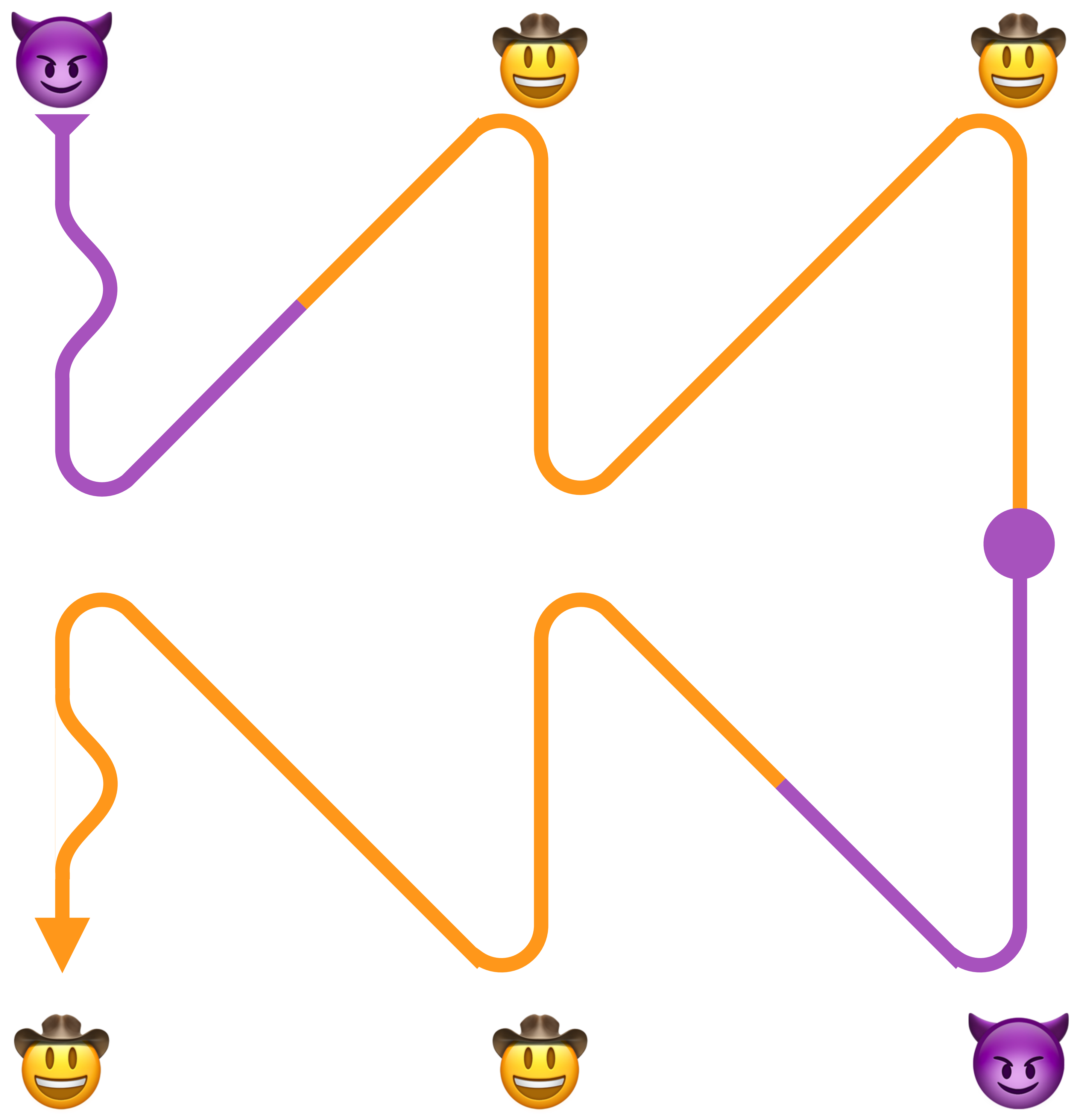
😎 Caller + 🤠 Called



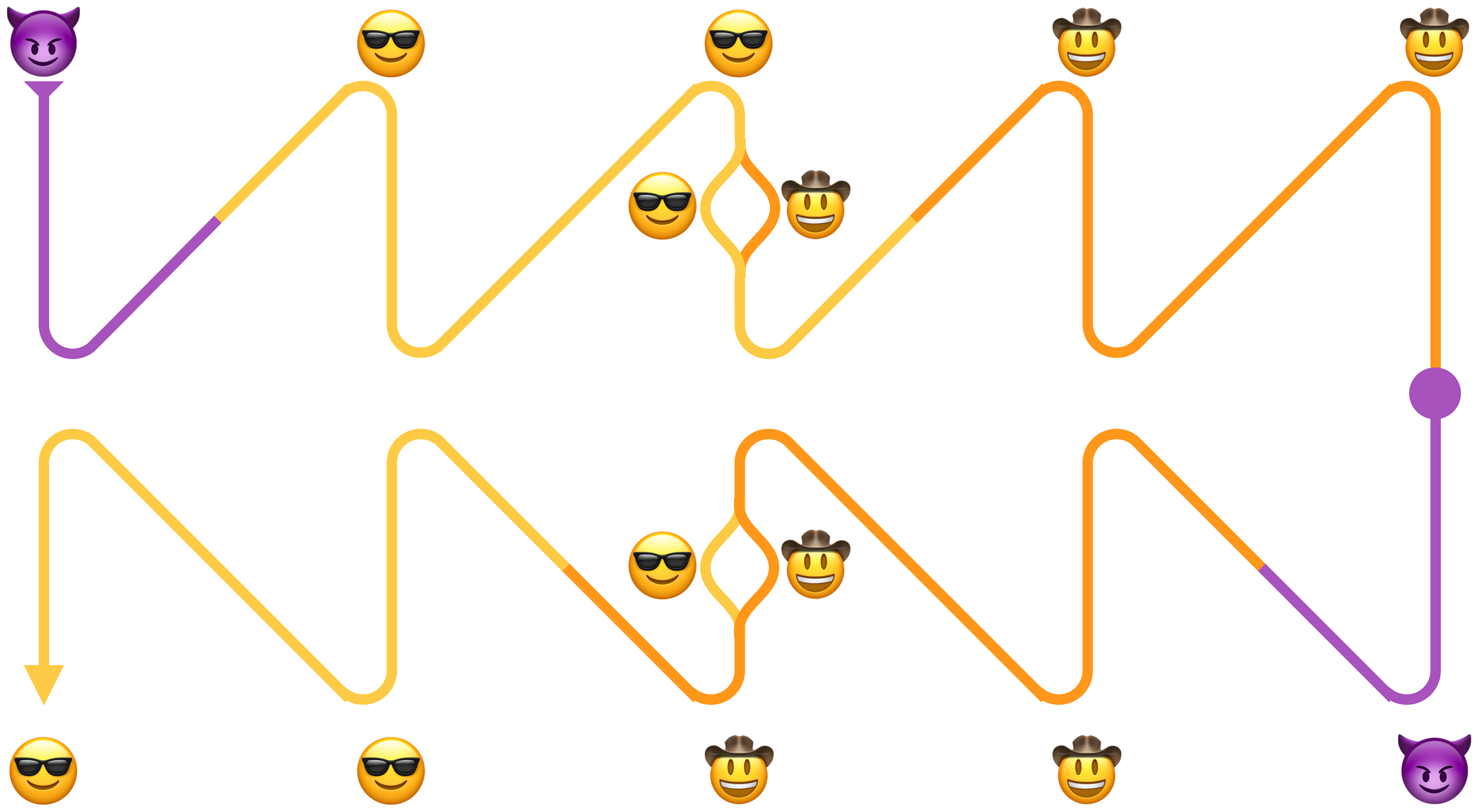
😎 Caller



🤠 Called



😎 Caller + 🤠 Called



One Definition Rule

Most named entities used in a program must be defined.
- Some may be defined only once (*e.g.*, non-inline functions).
- Others may be defined once per translation unit (*e.g.*, inline functions).

All the definitions of an entity must match.
- They must use the same sequence of tokens.
- They must generally refer to the same other entities.
- Sometimes they may refer to different constants with the same value.

Linkage name: *tokens, values, linkage names*

C: Cras mollis scelerisque nunc. Nullam arcu. Aliquam consequat. Curabitur augue lorem, dapibus quis, laoreet et, pretium ac, nisi. Aenean magna nisl, mollis quis, molestie eu, feugiat in, orci. In hac habitasse platea dictumst.

D: et risus vulputate vehicula. Donec lobortis risus a elit. Etiam tempor. Ut ullamcorper, ligula eu tempor congue, eros est euismod turpis, id tincidunt sapien risus a quam. Maecenas fermentum consequat mi. Donec

E: fermentum. Pellentesque malesuada nulla a mi. Duis sapien sem, aliquet nec, commodo eget, consequat quis, neque. Aliquam faucibus, elit ut dictum aliquet, felis nisl adipiscing sapien, sed malesuada diam lacus eget erat.

B: commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

D: et risus vulputate vehicula. Donec lobortis risus a elit. Etiam tempor. Ut ullamcorper, ligula eu tempor congue, eros est euismod turpis, id tincidunt sapien risus a quam. Maecenas fermentum consequat mi. Donec

A: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

C: Curabitur pretium tincidunt lacus. Nulla gravida orci a odio. Nullam varius, turpis et commodo pharetra, est eros bibendum elit, nec luctus magna felis sollicitudin mauris. Integer in mauris eu nibh euismod gravida. Duis ac tellus

D: et risus vulputate vehicula. Donec lobortis risus a elit. Etiam tempor. Ut ullamcorper, ligula eu tempor congue, eros est euismod turpis, id tincidunt sapien risus a quam. Maecenas fermentum consequat mi. Donec

E: fermentum. Pellentesque malesuada nulla a mi. Duis sapien sem, aliquet nec, commodo eget, consequat quis, neque. Aliquam faucibus, elit ut dictum aliquet, felis nisl adipiscing sapien, sed malesuada diam lacus eget erat.

A: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

C: Cras mollis scelerisque nunc. Nullam arcu. Aliquam consequat. Curabitur augue lorem, dapibus quis, laoreet et, pretium ac, nisi. Aenean magna nisl, mollis quis, molestie eu, feugiat in, orci. In hac habitasse platea dictumst.

A: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

B: commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

C: Curabitur pretium tincidunt lacus. Nulla gravida orci a odio. Nullam varius, turpis et commodo pharetra, est eros bibendum elit, nec luctus magna felis sollicitudin mauris. Integer in mauris eu nibh euismod gravida. Duis ac tellus

E: fermentum. Pellentesque malesuada nulla a mi. Duis sapien sem, aliquet nec, commodo eget, consequat quis, neque. Aliquam faucibus, elit ut dictum aliquet, felis nisl adipiscing sapien, sed malesuada diam lacus eget erat.

B: commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

C: Curabitur pretium tincidunt lacus. Nulla gravida orci a odio. Nullam varius, turpis et commodo pharetra, est eros bibendum elit, nec luctus magna felis sollicitudin mauris. Integer in mauris eu nibh euismod gravida. Duis ac tellus

E: fermentum. Pellentesque malesuada nulla a mi. Duis sapien sem, aliquet nec, commodo eget, consequat quis, neque. Aliquam faucibus, elit ut dictum aliquet, felis nisl adipiscing sapien, sed malesuada diam lacus eget erat.

A: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

A: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

A: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

B: commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

B: commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

B: commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

C: Cras mollis scelerisque nunc. Nullam arcu. Aliquam consequat. Curabitur augue lorem, dapibus quis, laoreet et, pretium ac, nisi. Aenean magna nisl, mollis quis, molestie eu, feugiat in, orci. In hac habitasse platea dictumst.

C: Cras mollis scelerisque nunc. Nullam arcu. Aliquam consequat. Curabitur augue lorem, dapibus quis, laoreet et, pretium ac, nisi. Aenean magna nisl, mollis quis, molestie eu, feugiat in, orci. In hac habitasse platea dictumst.

C: Curabitur pretium tincidunt lacus. Nulla gravida orci a odio. Nullam varius, turpis et commodo pharetra, est eros bibendum elit, nec luctus magna felis sollicitudin mauris. Integer in mauris eu nibh euismod gravida. Duis ac tellus

C: Curabitur pretium tincidunt lacus. Nulla gravida orci a odio. Nullam varius, turpis et commodo pharetra, est eros bibendum elit, nec luctus magna felis sollicitudin mauris. Integer in mauris eu nibh euismod gravida. Duis ac tellus

C: Curabitur pretium tincidunt lacus. Nulla gravida orci a odio. Nullam varius, turpis et commodo pharetra, est eros bibendum elit, nec luctus magna felis sollicitudin mauris. Integer in mauris eu nibh euismod gravida. Duis ac tellus

D: et risus vulputate vehicula. Donec lobortis risus a elit. Etiam tempor. Ut ullamcorper, ligula eu tempor congue, eros est euismod turpis, id tincidunt sapien risus a quam. Maecenas fermentum consequat mi. Donec

D: et risus vulputate vehicula. Donec lobortis risus a elit. Etiam tempor. Ut ullamcorper, ligula eu tempor congue, eros est euismod turpis, id tincidunt sapien risus a quam. Maecenas fermentum consequat mi. Donec

D: et risus vulputate vehicula. Donec lobortis risus a elit. Etiam tempor. Ut ullamcorper, ligula eu tempor congue, eros est euismod turpis, id tincidunt sapien risus a quam. Maecenas fermentum consequat mi. Donec

E: fermentum. Pellentesque malesuada nulla a mi. Duis sapien sem, aliquet nec, commodo eget, consequat quis, neque. Aliquam faucibus, elit ut dictum aliquet, felis nisl adipiscing sapien, sed malesuada diam lacus eget erat.

E: fermentum. Pellentesque malesuada nulla a mi. Duis sapien sem, aliquet nec, commodo eget, consequat quis, neque. Aliquam faucibus, elit ut dictum aliquet, felis nisl adipiscing sapien, sed malesuada diam lacus eget erat.

E: fermentum. Pellentesque malesuada nulla a mi. Duis sapien sem, aliquet nec, commodo eget, consequat quis, neque. Aliquam faucibus, elit ut dictum aliquet, felis nisl adipiscing sapien, sed malesuada diam lacus eget erat.

E: fermentum. Pellentesque malesuada nulla a mi. Duis sapien sem, aliquet nec, commodo eget, consequat quis, neque. Aliquam faucibus, elit ut dictum aliquet, felis nisl adipiscing sapien, sed malesuada diam lacus eget erat.

A: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

B: commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

🔥

C: Cras mollis scelerisque nunc. Nullam arcu. Aliquam consequat. Curabitur augue lorem, dapibus quis, laoreet et, pretium ac, nisi. Aenean magna nisl, mollis quis, molestie eu, feugiat in, orci. In hac habitasse platea dictumst.

C: Curabitur pretium tincidunt lacus. Nulla gravida orci a odio. Nullam varius, turpis et commodo pharetra, est eros bibendum elit, nec luctus magna felis sollicitudin mauris. Integer in mauris eu nibh euismod gravida. Duis ac tellus

D: et risus vulputate vehicula. Donec lobortis risus a elit. Etiam tempor. Ut ullamcorper, ligula eu tempor congue, eros est euismod turpis, id tincidunt sapien risus a quam. Maecenas fermentum consequat mi. Donec

E: fermentum. Pellentesque malesuada nulla a mi. Duis sapien sem, aliquet nec, commodo eget, consequat quis, neque. Aliquam faucibus, elit ut dictum aliquet, felis nisl adipiscing sapien, sed malesuada diam lacus eget erat.

Local phase

Validate small neighborhoods with high-complexity algorithms.
Describe neighborhood interactions coarsely in small tables.

Global phase

Collate the small tables into a big table of interactions.
Validate the interactions with low-complexity algorithms.

Indirect caller’s interface

fits many functions

```
int Nondecreasing( const int x )
interface
  {
   claim usable( x );


   implementation;

   claim x <= result;

   claim usable( x );
   claim usable( result );
  }
```

Function interface

fits a specific function

```
int Identity( const int x )
interface
  {
   claim usable( x );


   implementation;

   claim x == result;

   claim usable( x );
   claim usable( result );
  }
```

Indirect caller's interface

fits many specific interfaces

```
int Nondecreasing( const int x )
interface ->
  {
   claim usable( x );


   implementation;

   claim x <= result;

   claim usable( x );
   claim usable( result );
  }
```

Interface adapter

Similar to a template; instantiated by attaching it to the front of a function, creating a new function.

Nondecreasing becomes a type name, meaning "function fronted by the Nondecreasing adapter."

Nondecreasing

interface adapter

```
int Nondecreasing( const int x )
interface ->
  {
   claim usable( x );


   implementation;

   claim x <= result;

   claim usable( x );
   claim usable( result );
  }
```

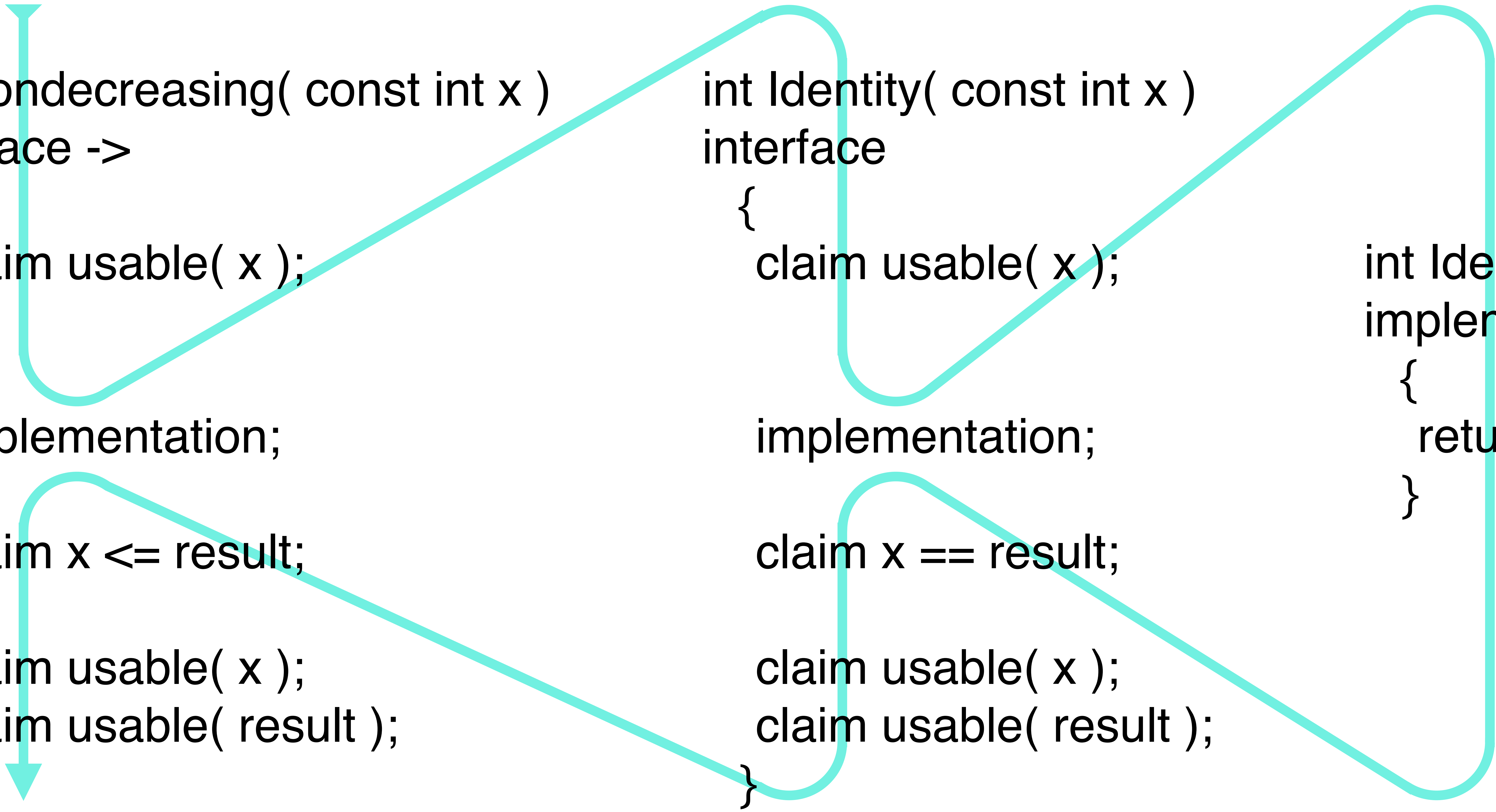
Identity

function interface

```
int Identity( const int x )
interface
  {
   claim usable( x );


   implementation;

   claim x == result;

   claim usable( x );
   claim usable( result );
  }
```

Nondecreasing -> Identity

"Nondecreasing fronting Identity"

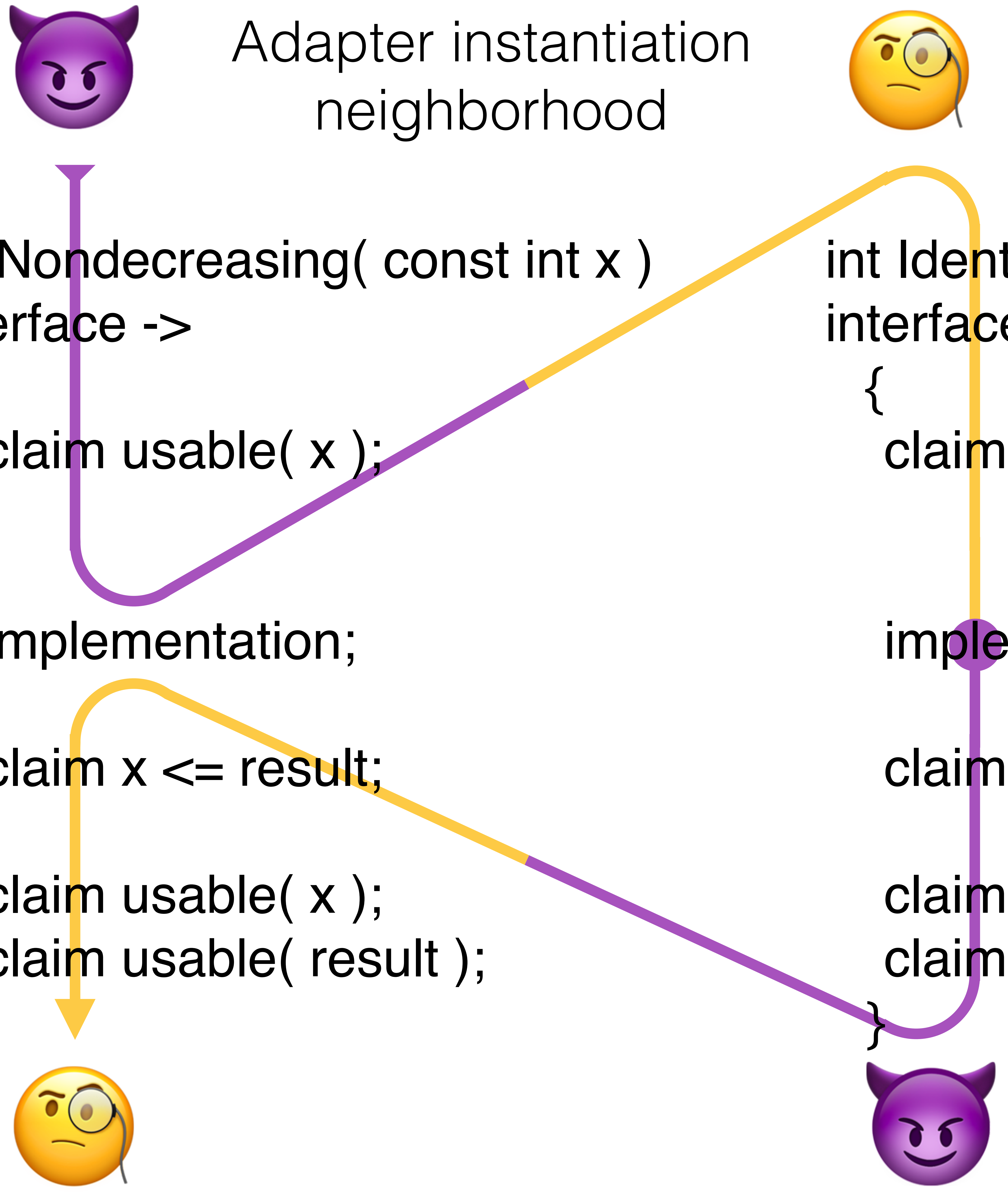


```
int Nondecreasing( const int x )
interface ->
  {
   claim usable( x );


   implementation;

   claim x <= result;

   claim usable( x );
   claim usable( result );
  }
```

```
int Identity( const int x )
interface
  {
   claim usable( x );


   implementation;

   claim x == result;

   claim usable( x );
   claim usable( result );
  }
```

```
int Identity( const int x )
implementation
  {
   return x;
  }
```

Adapter instantiation neighborhood

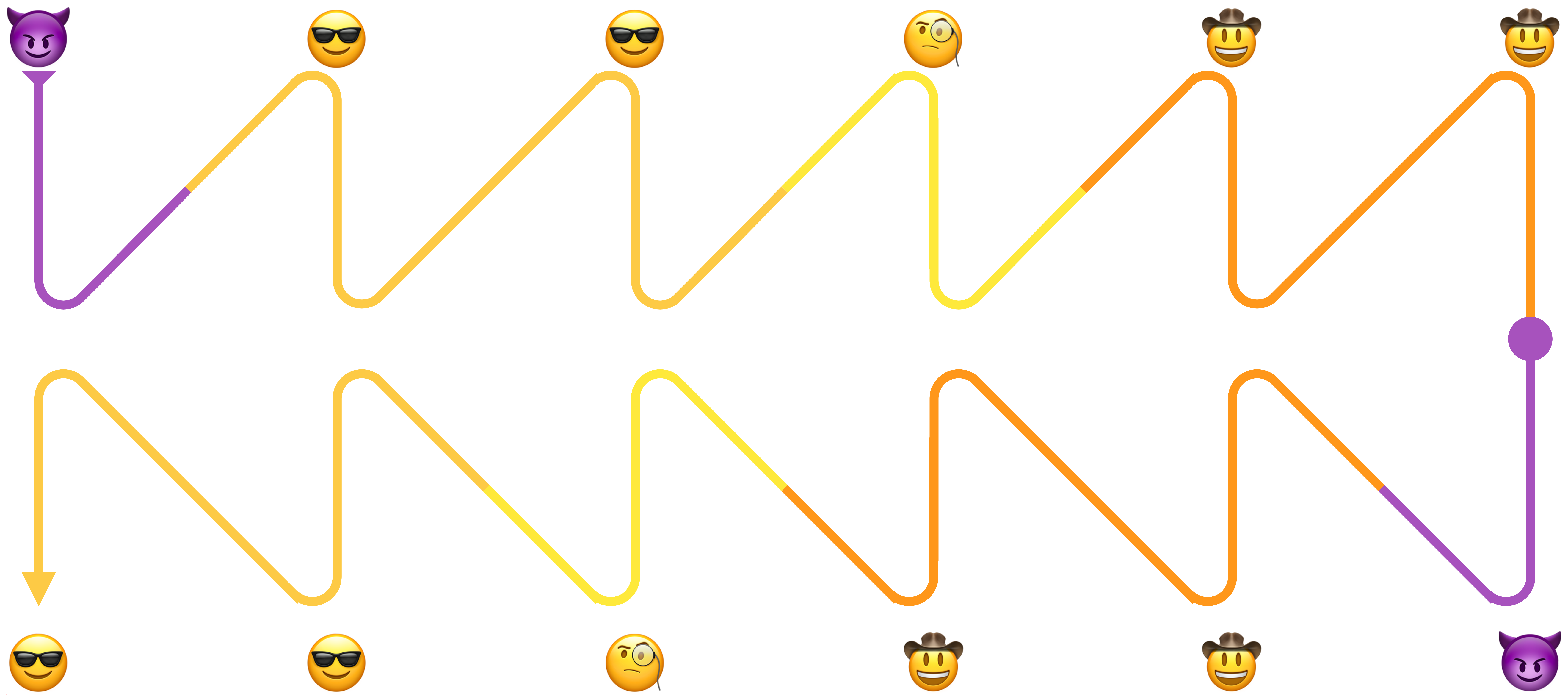


```
int Nondecreasing( const int x )
interface ->
  {
    claim usable( x );


    implementation;

    claim x <= result;

    claim usable( x );
    claim usable( result );
  }
```

```
int Identity( const int x )
interface
  {
    claim usable( x );


    implementation;

    claim x == result;

    claim usable( x );
    claim usable( result );
  }
```

😎 Caller 🧐 Instantiation 🤠 Called

😎 Caller + 🧐 Instantiation + 🤠 Called



```
int function_adapter( int x )
interface ->
  {
   // ...
  }
```

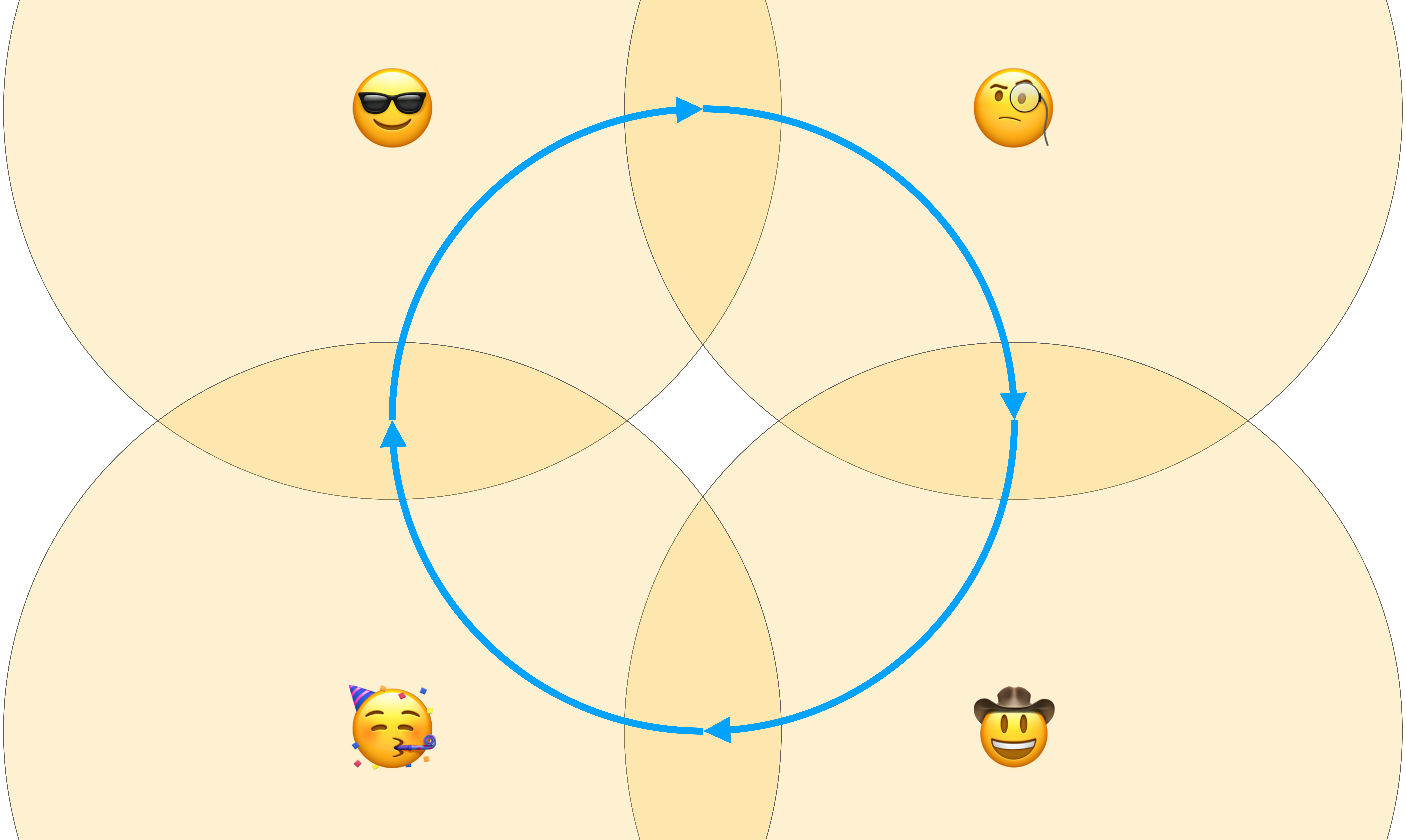
For indirect calls through pointers or references to functions

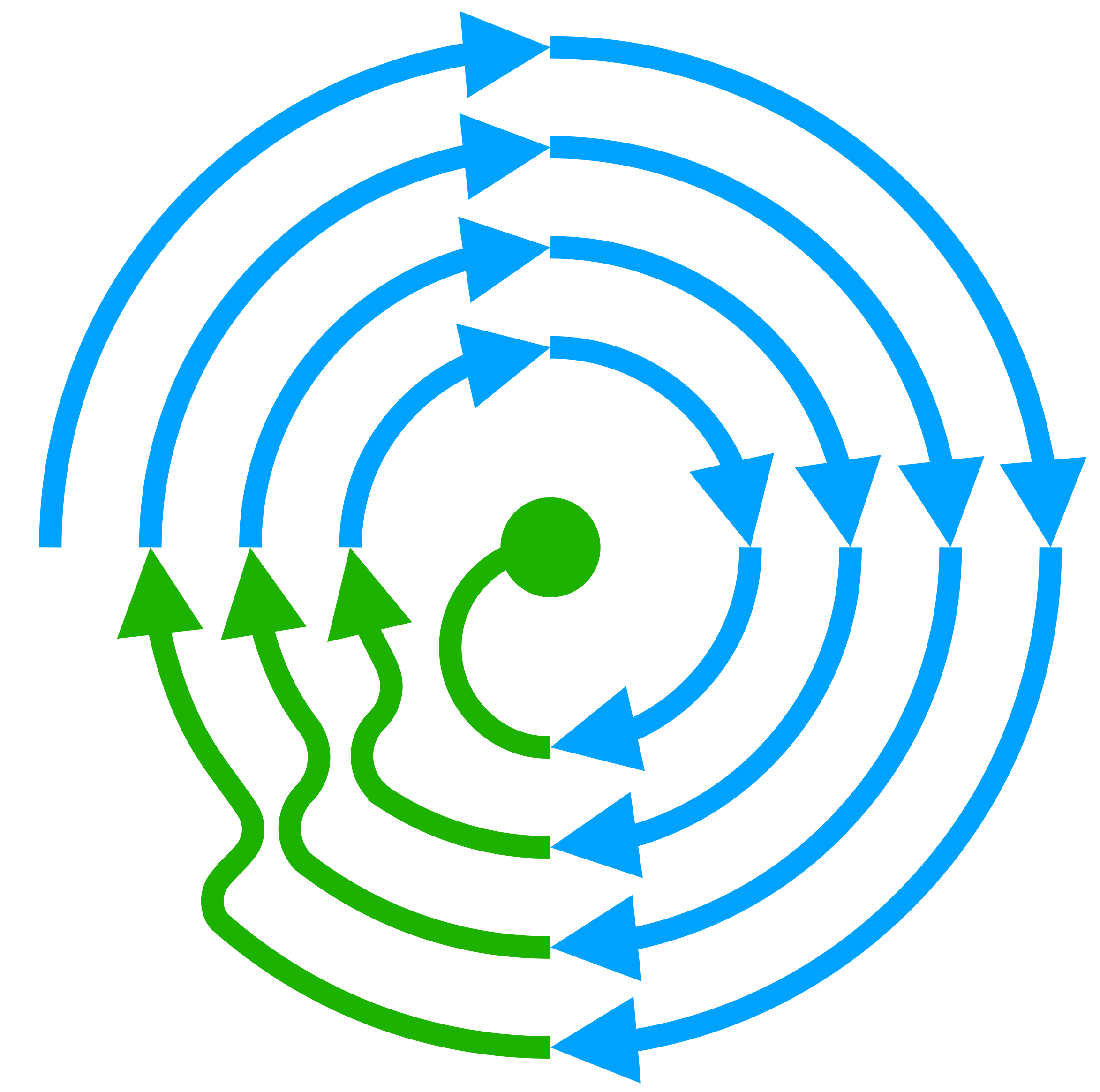
```
int virtual_function_adapter( int x )
interface virtual ->
  {
   // ...
  }
```

For indirect calls through virtual dispatch

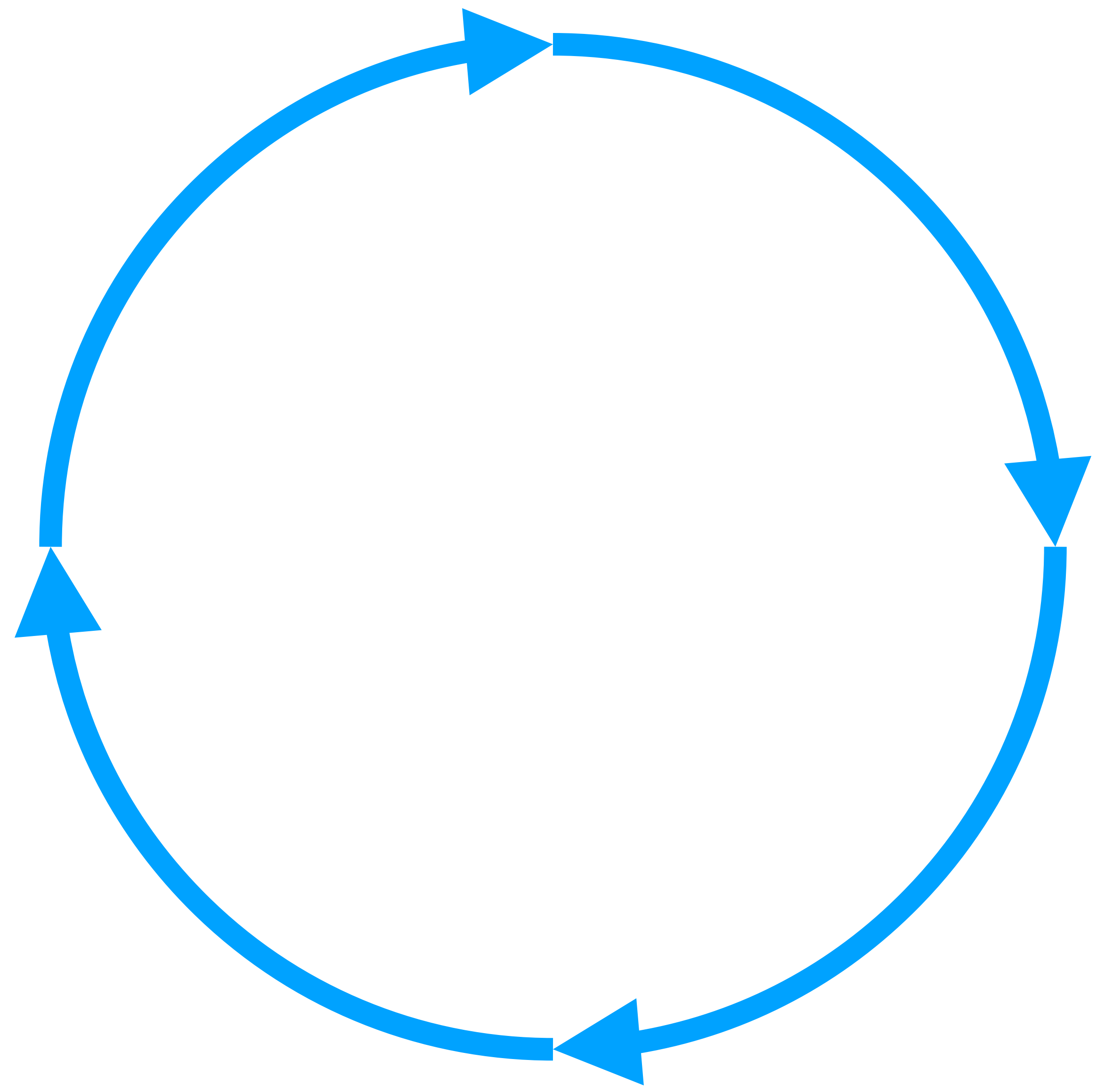
```
int member_function_adapter( int x )
interface type ->
  {
   // ...
  }
```

For indirect calls through pointers to member functions

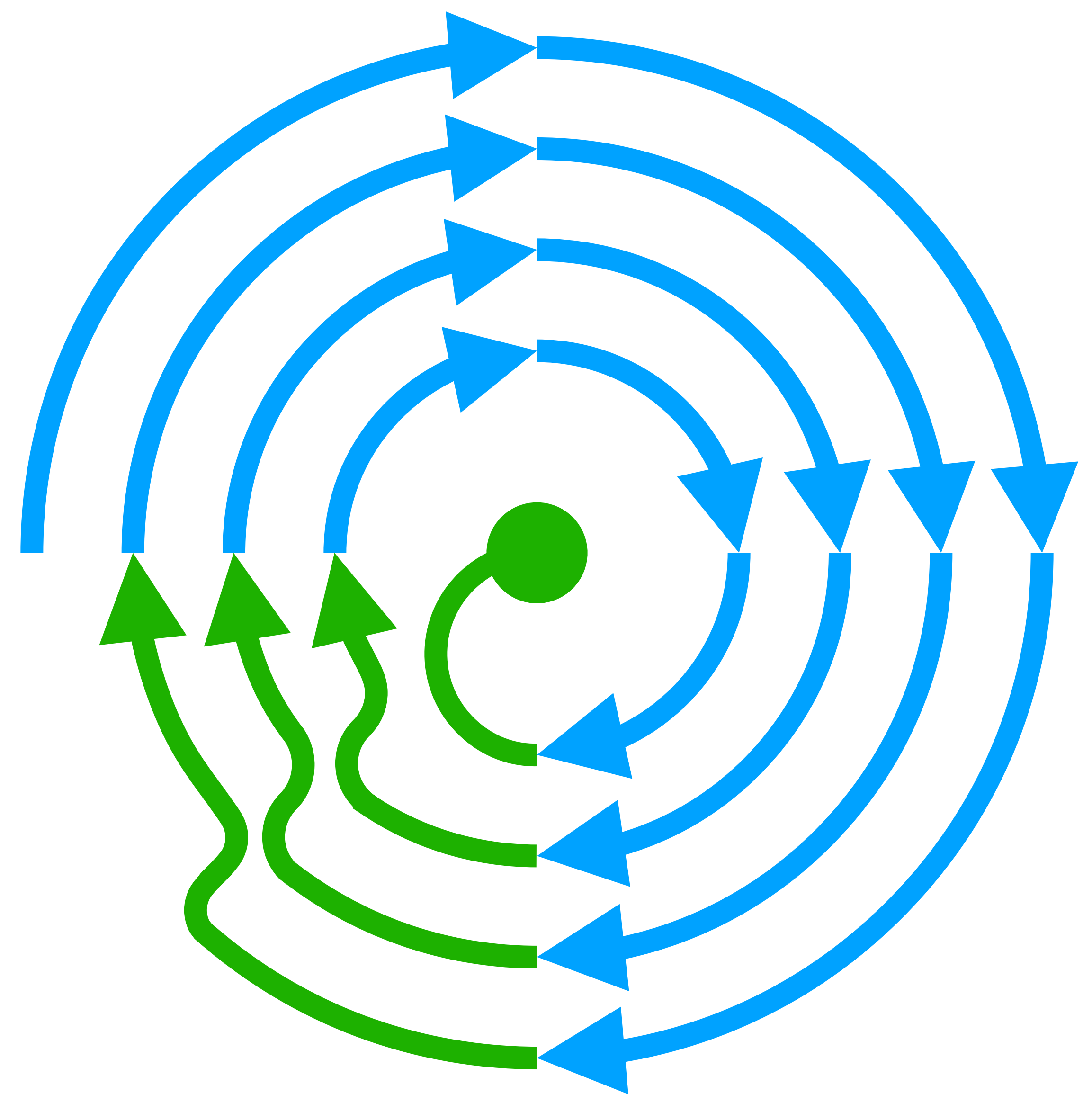




Bounded Recursion



Unbounded Recursion

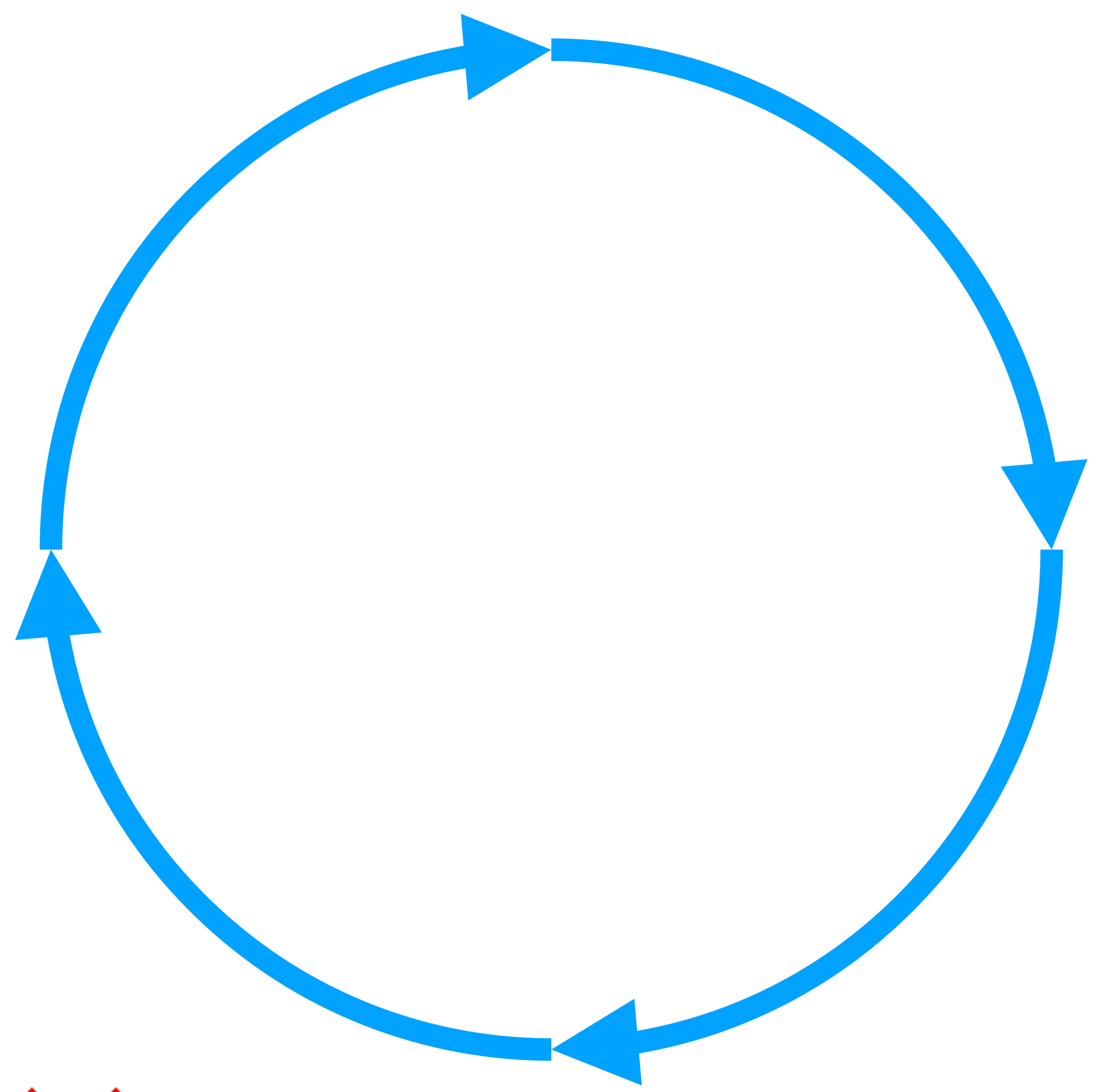


Recursion Bound

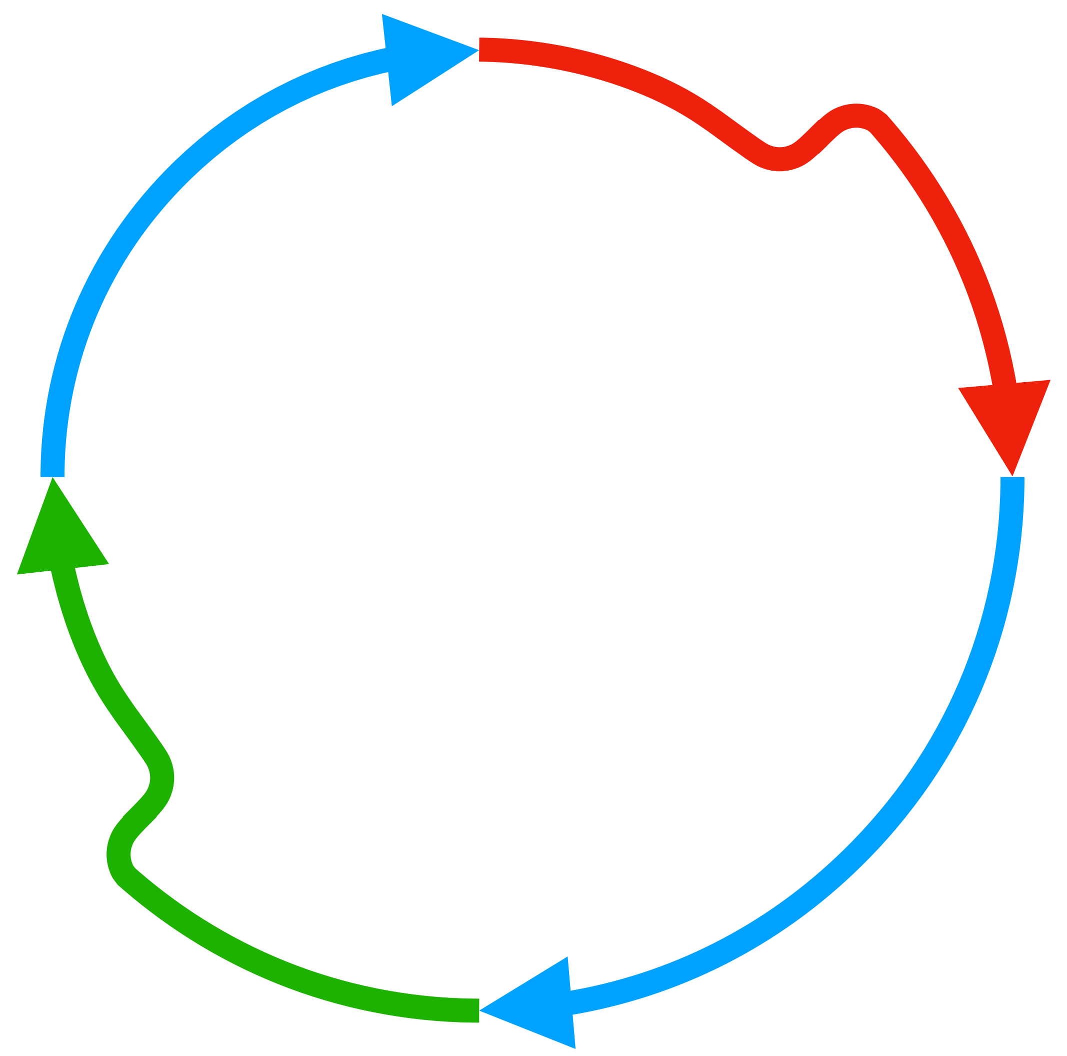
✅ Decreases somewhere in every cycle

✅ Never increases within a cycle

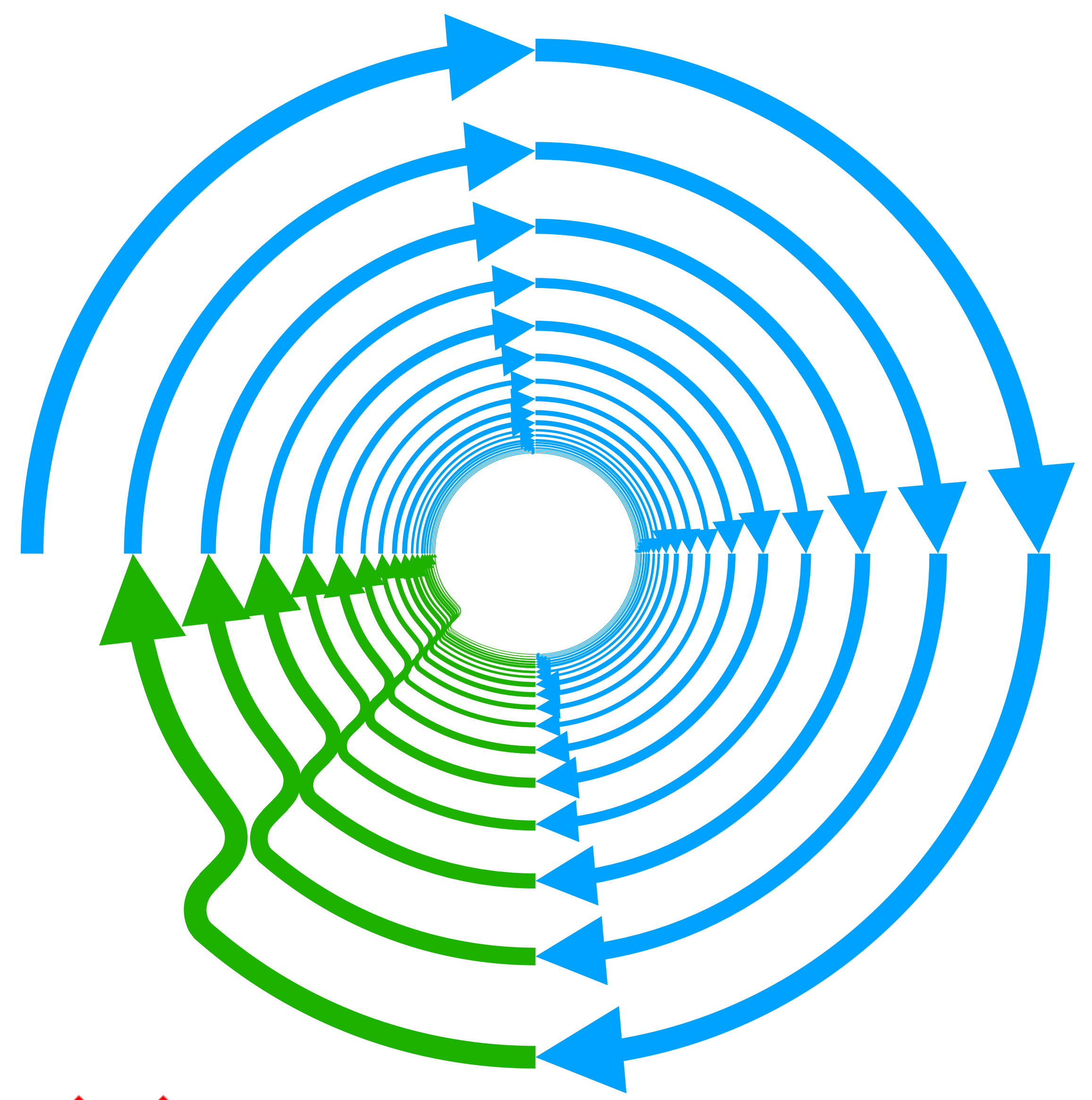
✅ Eventually runs out



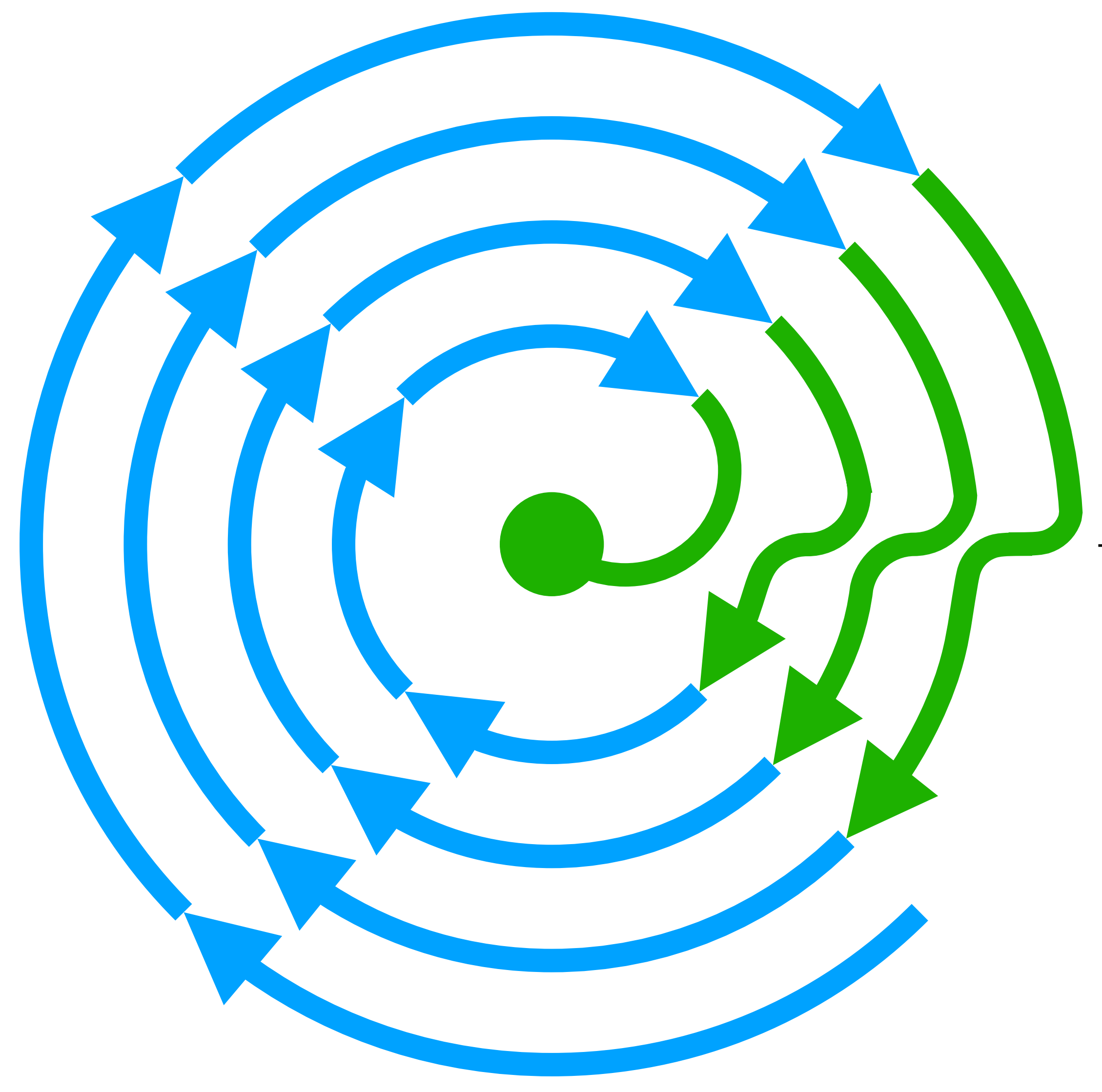
❌ Never decreases



❌ Sometimes increases

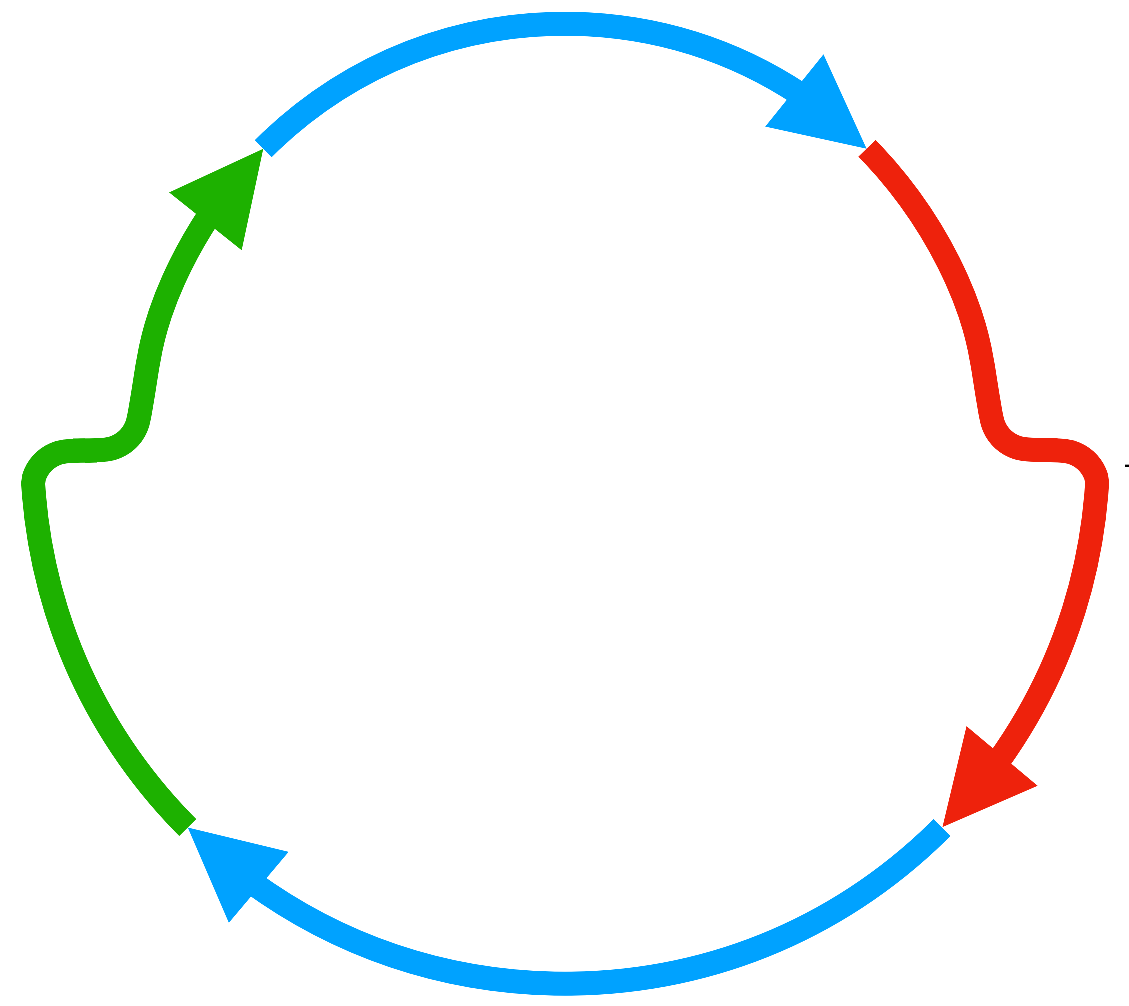


❌ Never runs out



Steps that always decrease the recursion bound

(Every cycle needs one!)



Steps that may increase the recursion bound

(Not allowed in cycles!)

Local phase

Ensure recursion bounds eventually run out.

Compile lists of all steps connecting interfaces.

Mark possibly-increasing steps with 🚫.

Mark necessarily-decreasing steps with ✅.

Global phase

Check that no cycle contains step marked 🚫.

Check that every cycle contains a step marked ✅.

Direct Function Call

Interface ➞ Implementation ➞ Interface
Interface ➞ Interface
Adapter ➞ Interface

Indirect Function Call

Interface ➞ Implementation ➞ Adapter
Interface ➞ Adapter
Adapter ➞ Adapter

Adapter Implementation Point

Adapter ➞ Interface
Adapter ➞ Adapter

b ➞ h 🚫	a ➞ b 🚫
c ➞ d	a ➞ i 🚫
d ➞ b	c ➞ d
g ➞ c ✓	d ➞ b
	f ➞ g

b ➞ c	b ➞ c
b ➞ f	c ➞ d
b ➞ h 🚫	g ➞ c ✓
c ➞ e	h ➞ g
i ➞ i ✓	f ➞ h

Direct Function Call

- Interface → Implementation → Interface
- Interface → Interface
- Adapter → Interface

Indirect Function Call

- Interface → Implementation → Adapter
- Interface → Adapter
- Adapter → Adapter

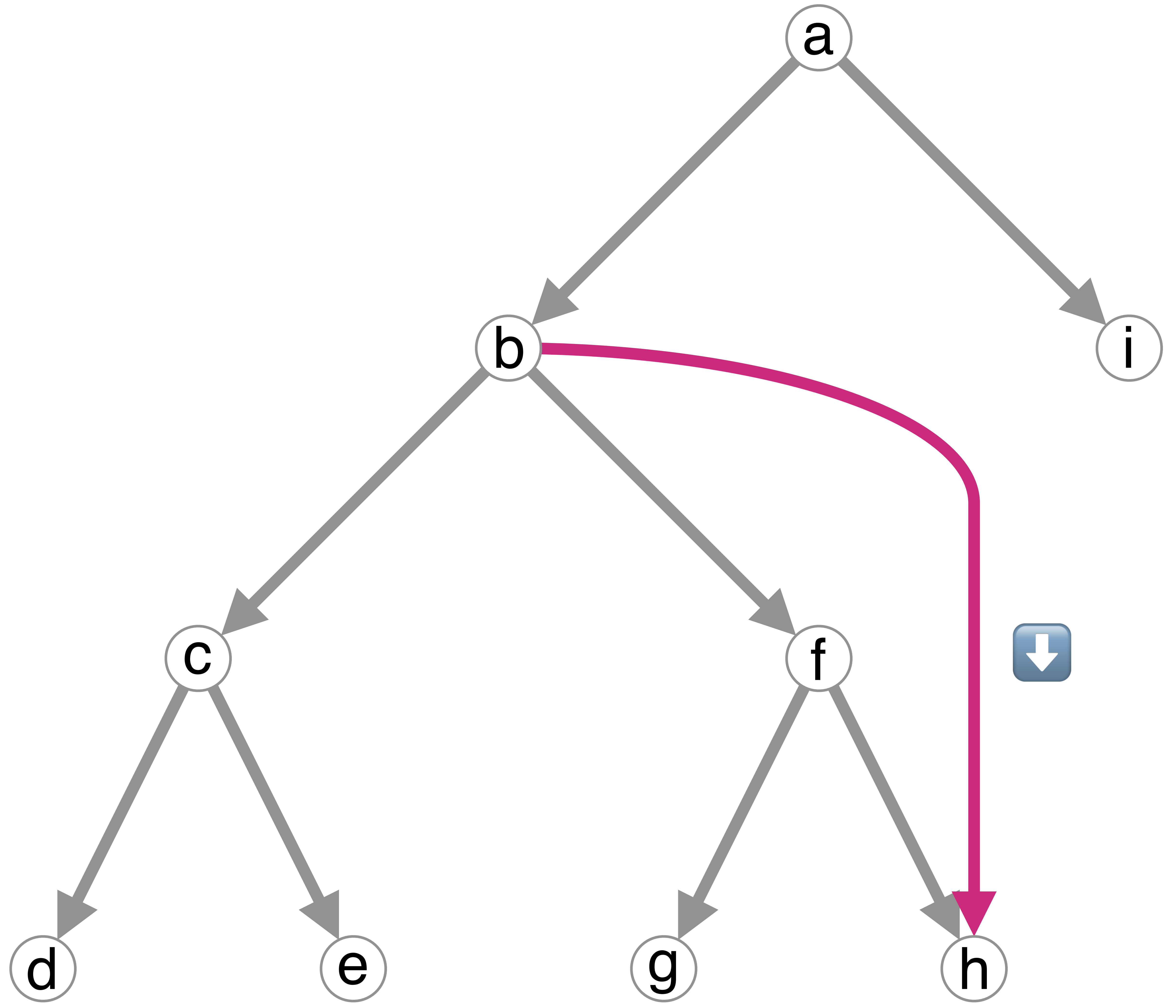
Adapter Implementation Point

- Adapter → Interface
- Adapter → Adapter

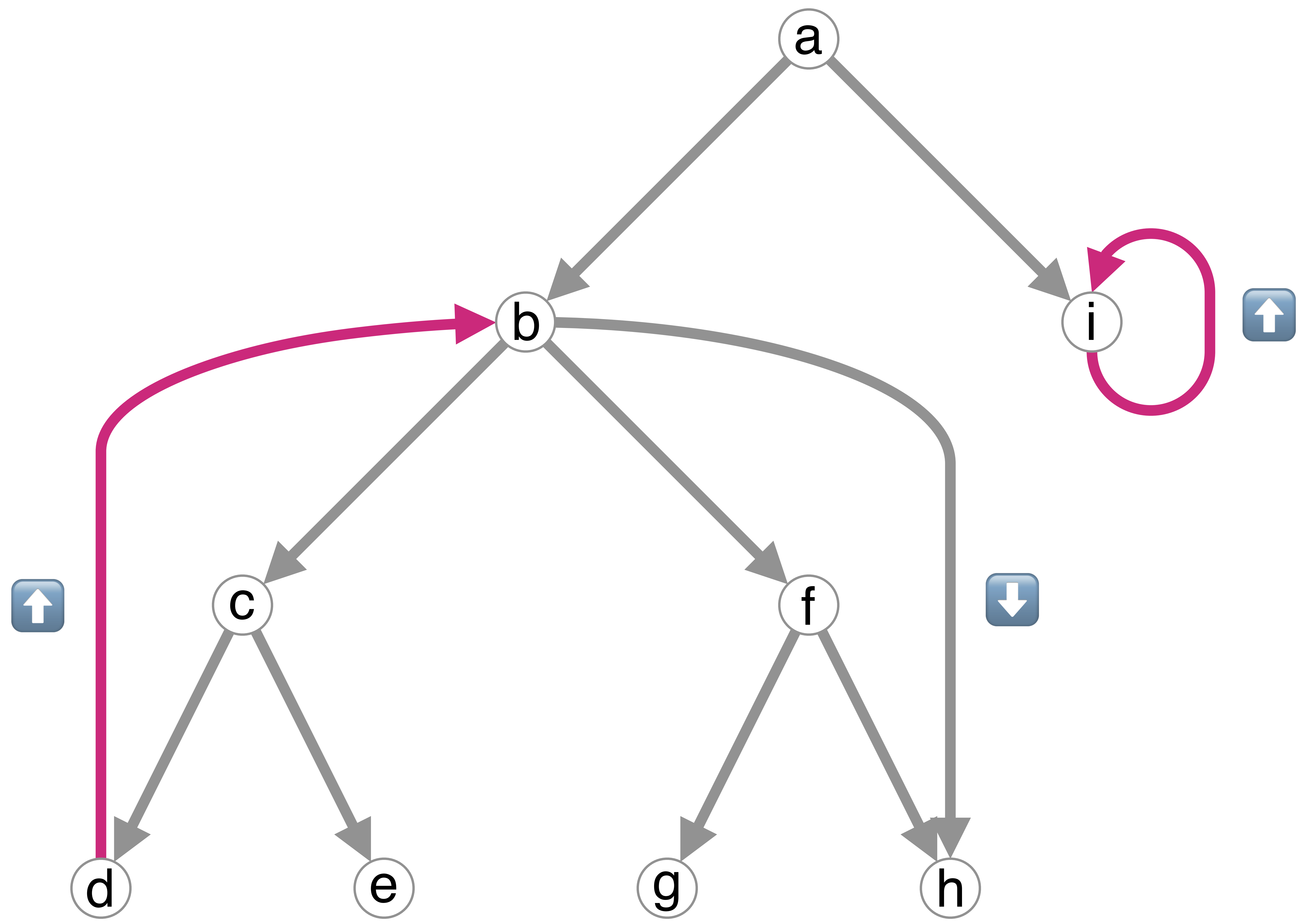
a → b	🚫
a → i	🚫
b → c	
b → f	
b → h	🚫
c → d	
c → e	
d → b	
f → g	
f → h	
g → c	✅
h → g	
i → i	✅

$$\begin{tikzcd}
 & & & & & a \arrow[dll] \arrow[drr] & & \\
 & & & b \arrow[dll] \arrow[drr] & & & & i \\
 & c \arrow[dl] \arrow[dr] & & & & f \arrow[dl] \arrow[dr] & & \\
d & & e & & g & & h &
\end{tikzcd}$$

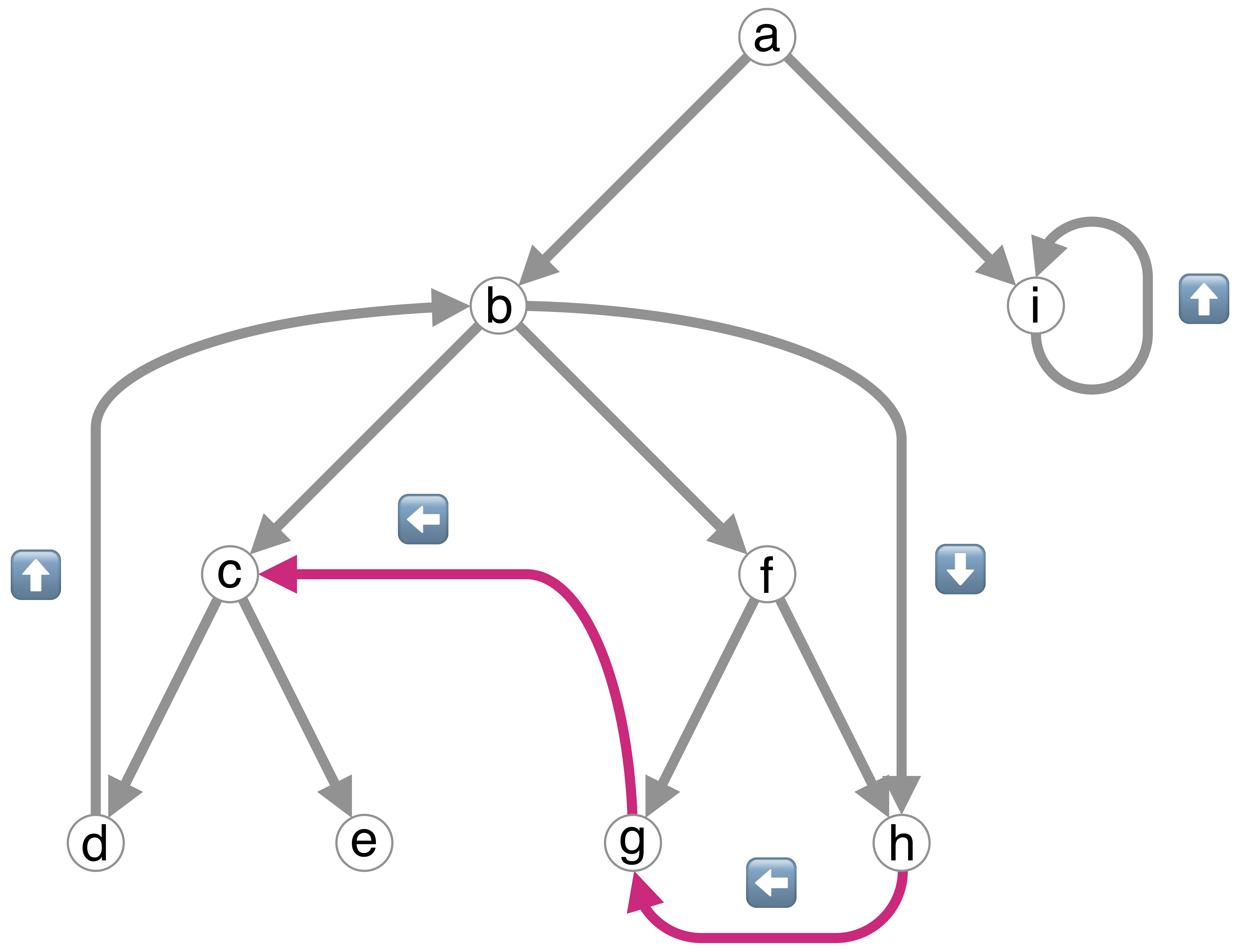
✅	$a \rightarrow b$	🚫
✅	$a \rightarrow i$	🚫
✅	$b \rightarrow c$	
✅	$b \rightarrow f$	
	$b \rightarrow h$	🚫
✅	$c \rightarrow d$	
✅	$c \rightarrow e$	
	$d \rightarrow b$	
✅	$f \rightarrow g$	
✅	$f \rightarrow h$	
	$g \rightarrow c$	🔄
	$h \rightarrow g$	
	$i \rightarrow i$	🔄



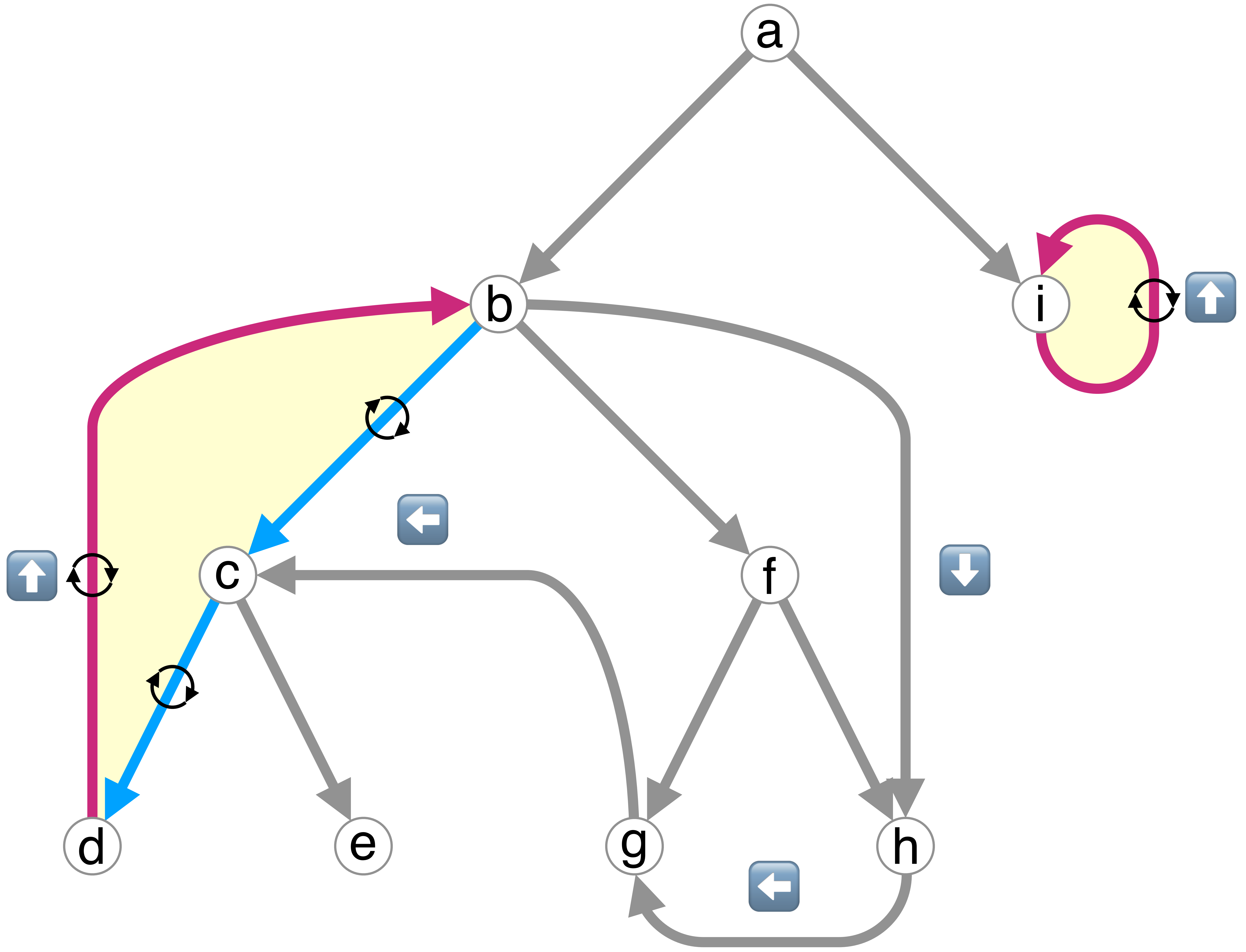
✅	a → b	🚫
✅	a → i	🚫
✅	b → c	
✅	b → f	
⬇️	b → h	🚫
✅	c → d	
✅	c → e	
	d → b	
✅	f → g	
✅	f → h	
	g → c	🔄
	h → g	
	i → i	🔄

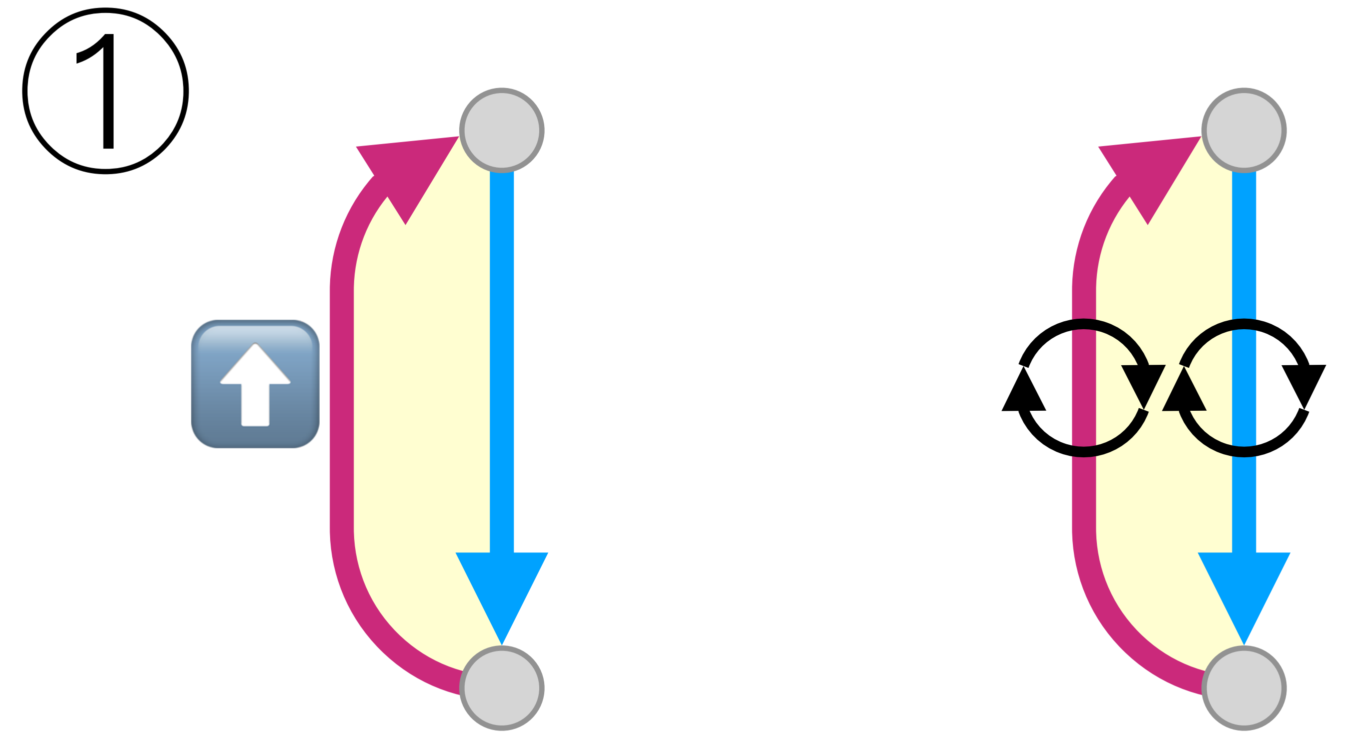


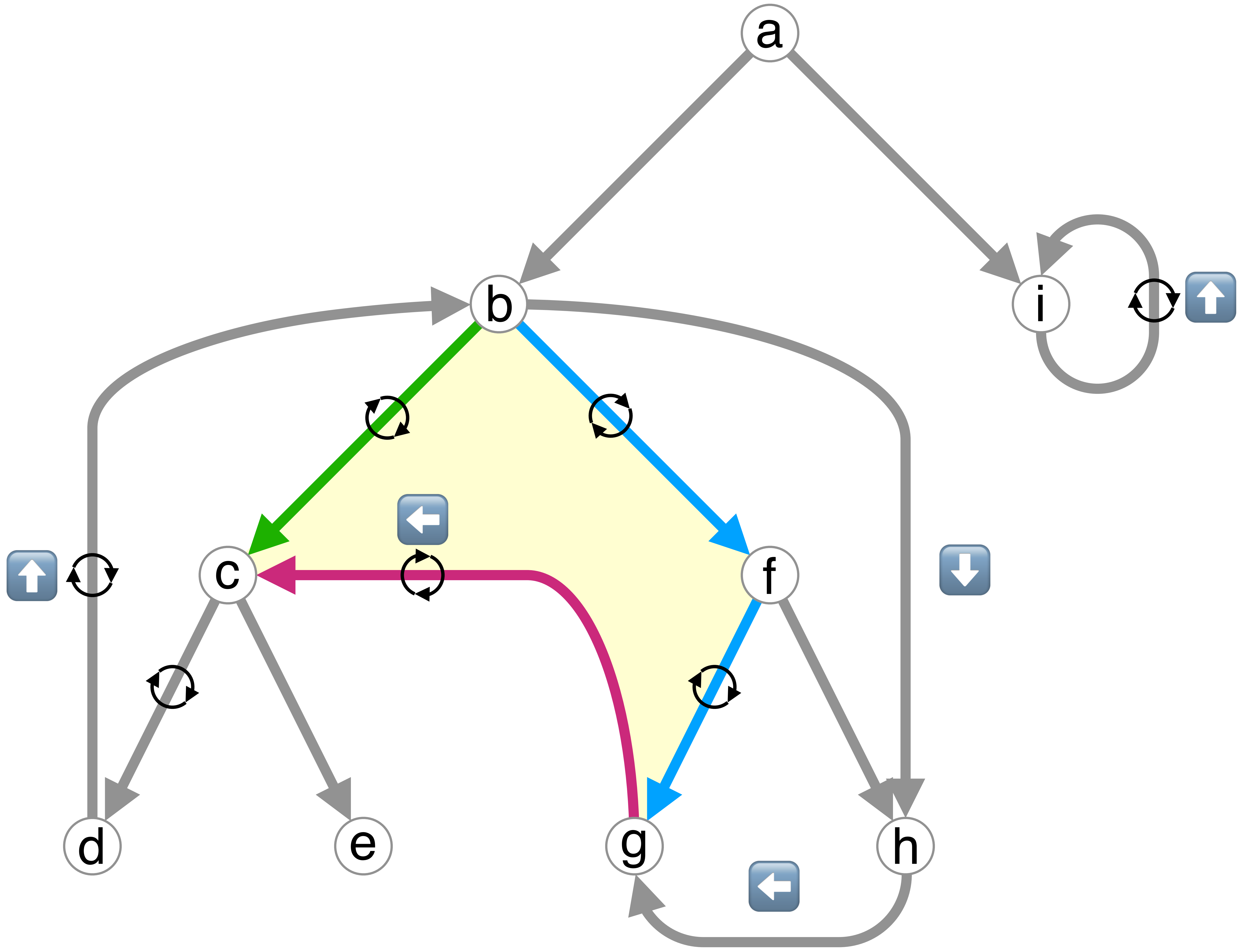
✅	a ➞ b	⊘
✅	a ➞ i	⊘
✅	b ➞ c	
✅	b ➞ f	
⬇️	b ➞ h	⊘
✅	c ➞ d	
✅	c ➞ e	
⬆️	d ➞ b	
✅	f ➞ g	
✅	f ➞ h	
	g ➞ c	⟳
	h ➞ g	
⬆️	i ➞ i	⟳

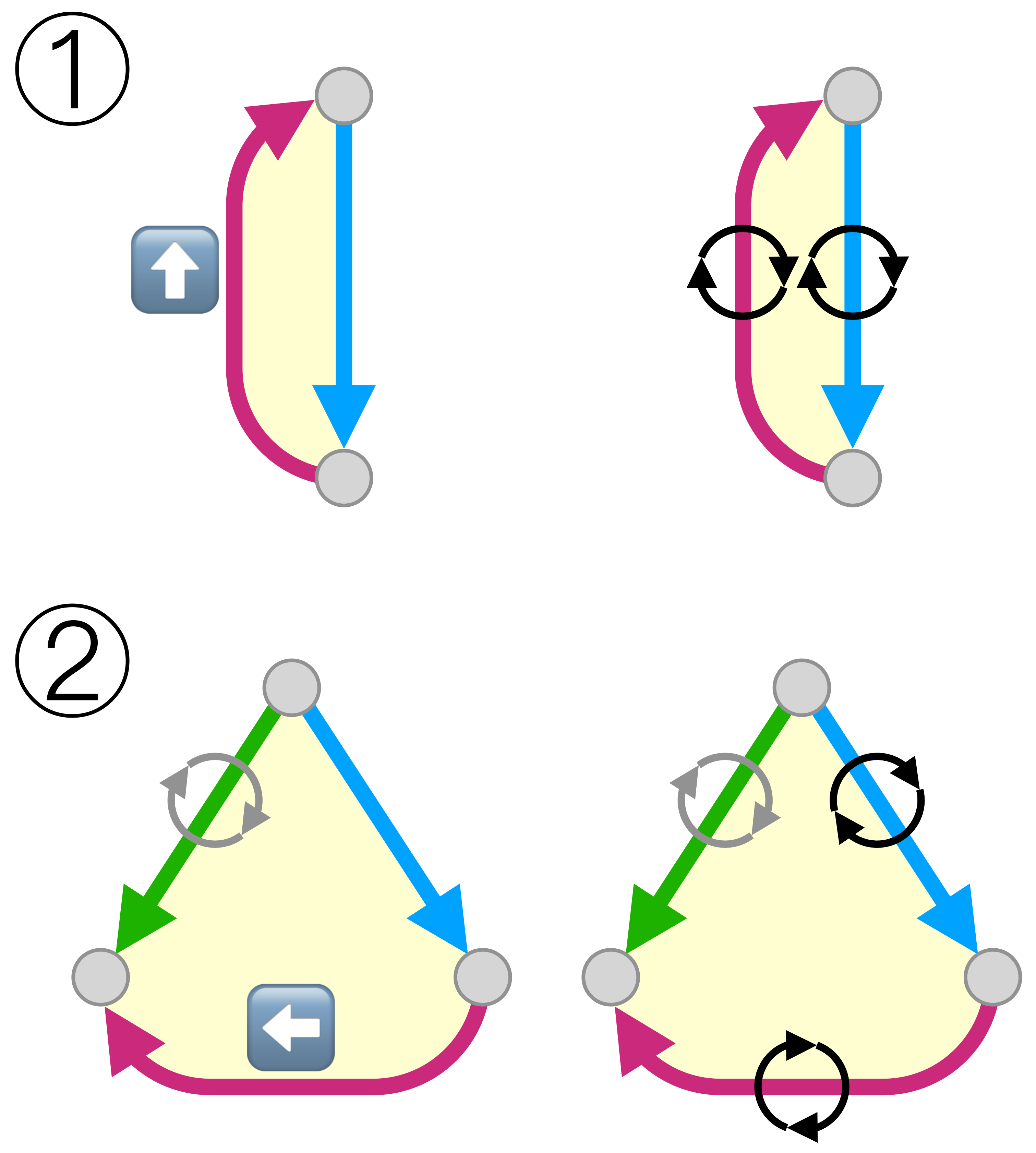


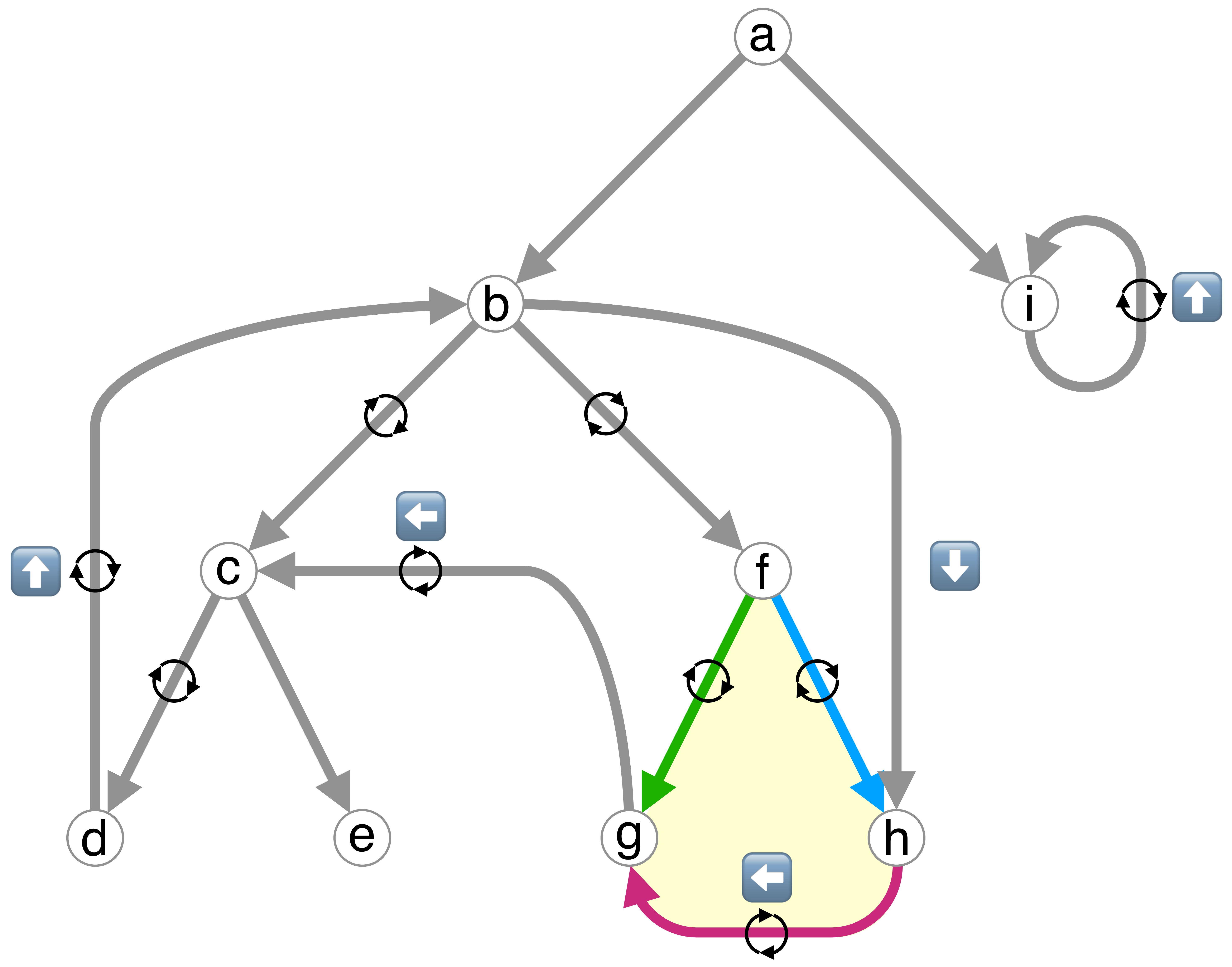
✅	a → b	🚫
✅	a → i	🚫
✅	b → c	
✅	b → f	
⬇️	b → h	🚫
✅	c → d	
✅	c → e	
⬆️	d → b	
✅	f → g	
✅	f → h	
⬅️	g → c	🔄
⬅️	h → g	
⬆️	i → i	🔄


a
b
c
d
e
f
g
h
i

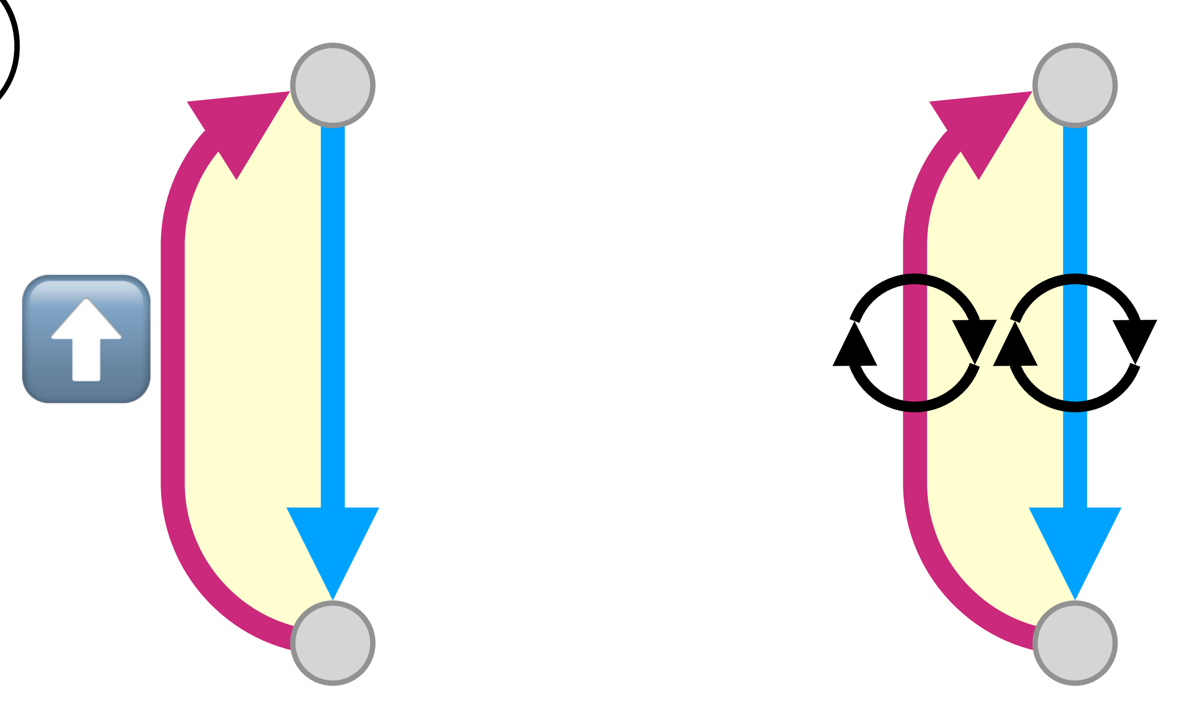

①


a
b
c
d
e
f
g
h
i

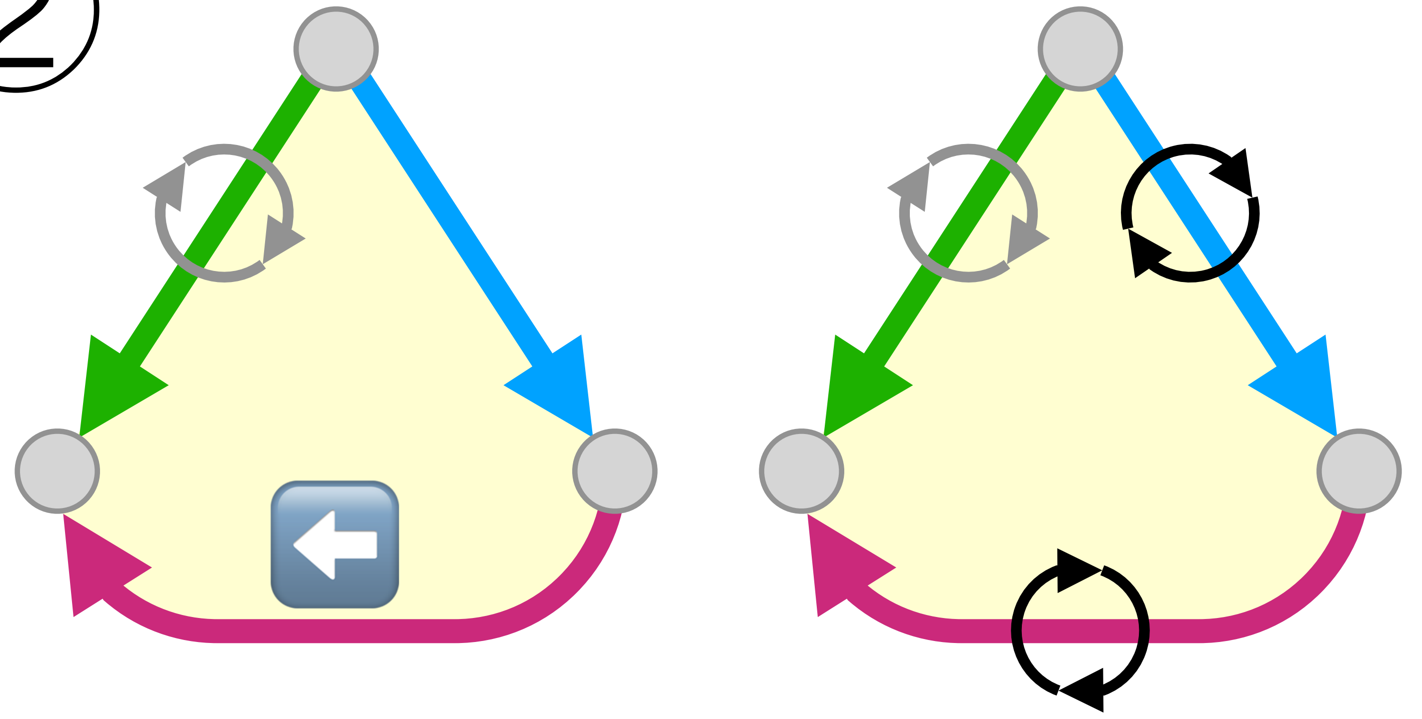

①
②

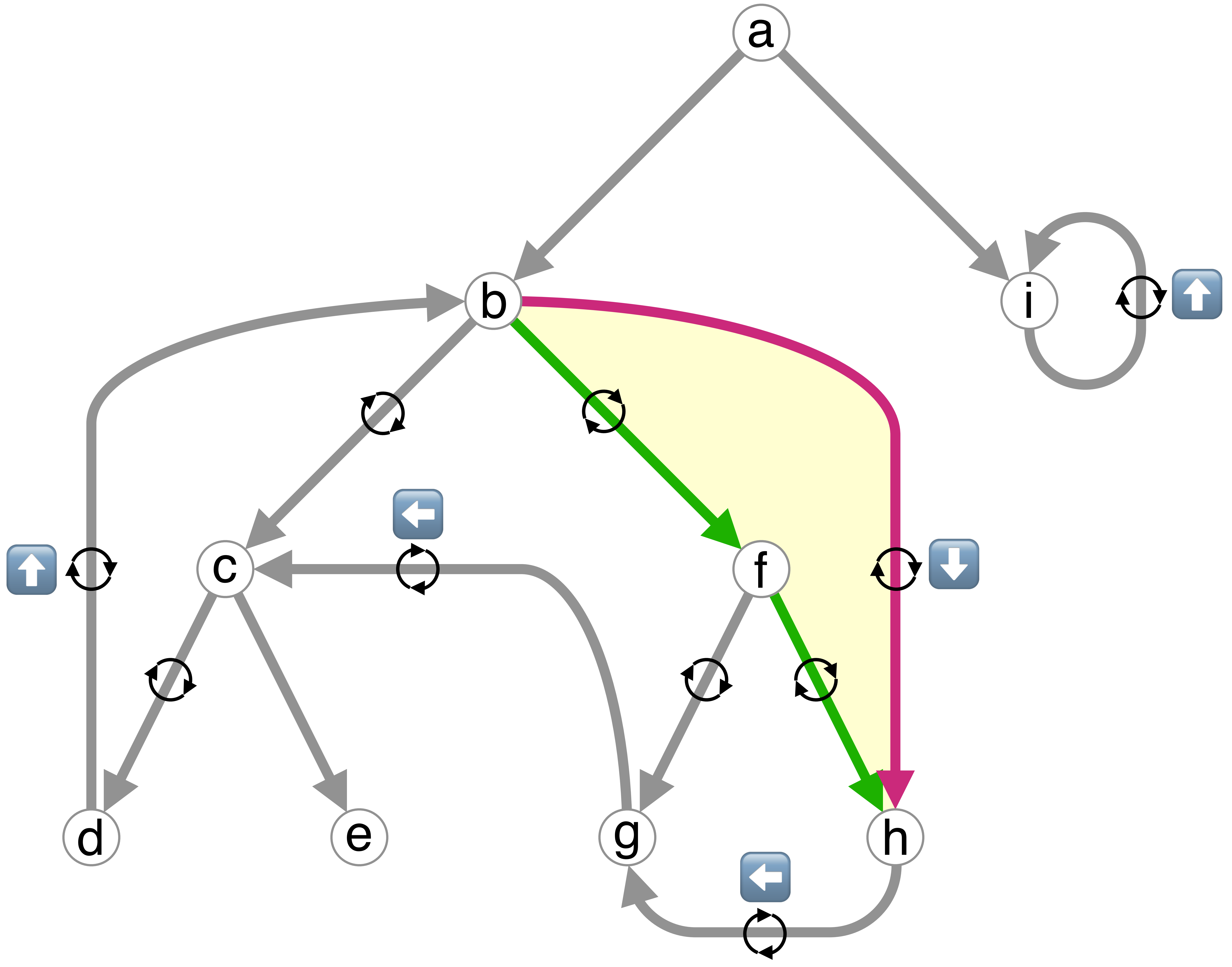


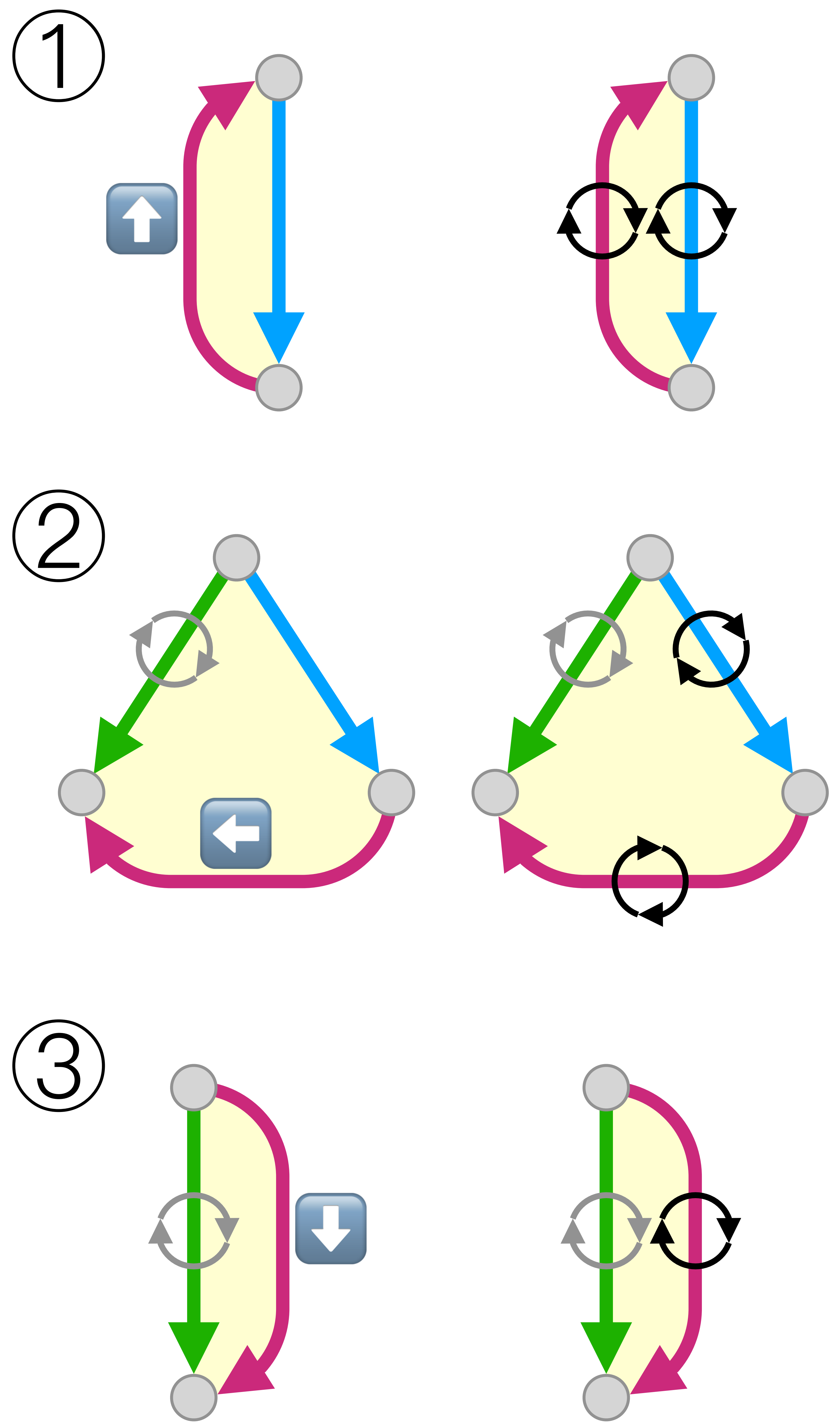
①

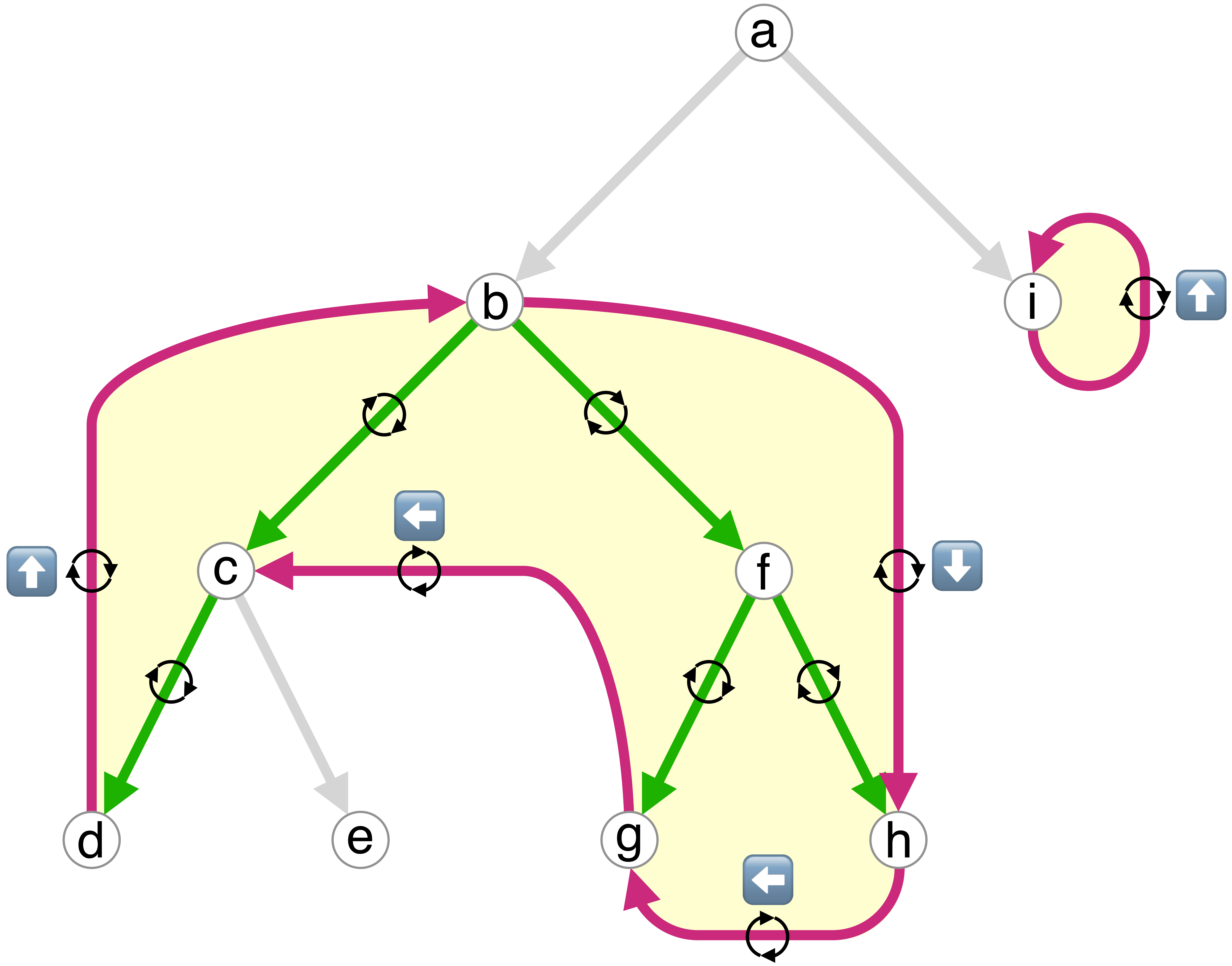


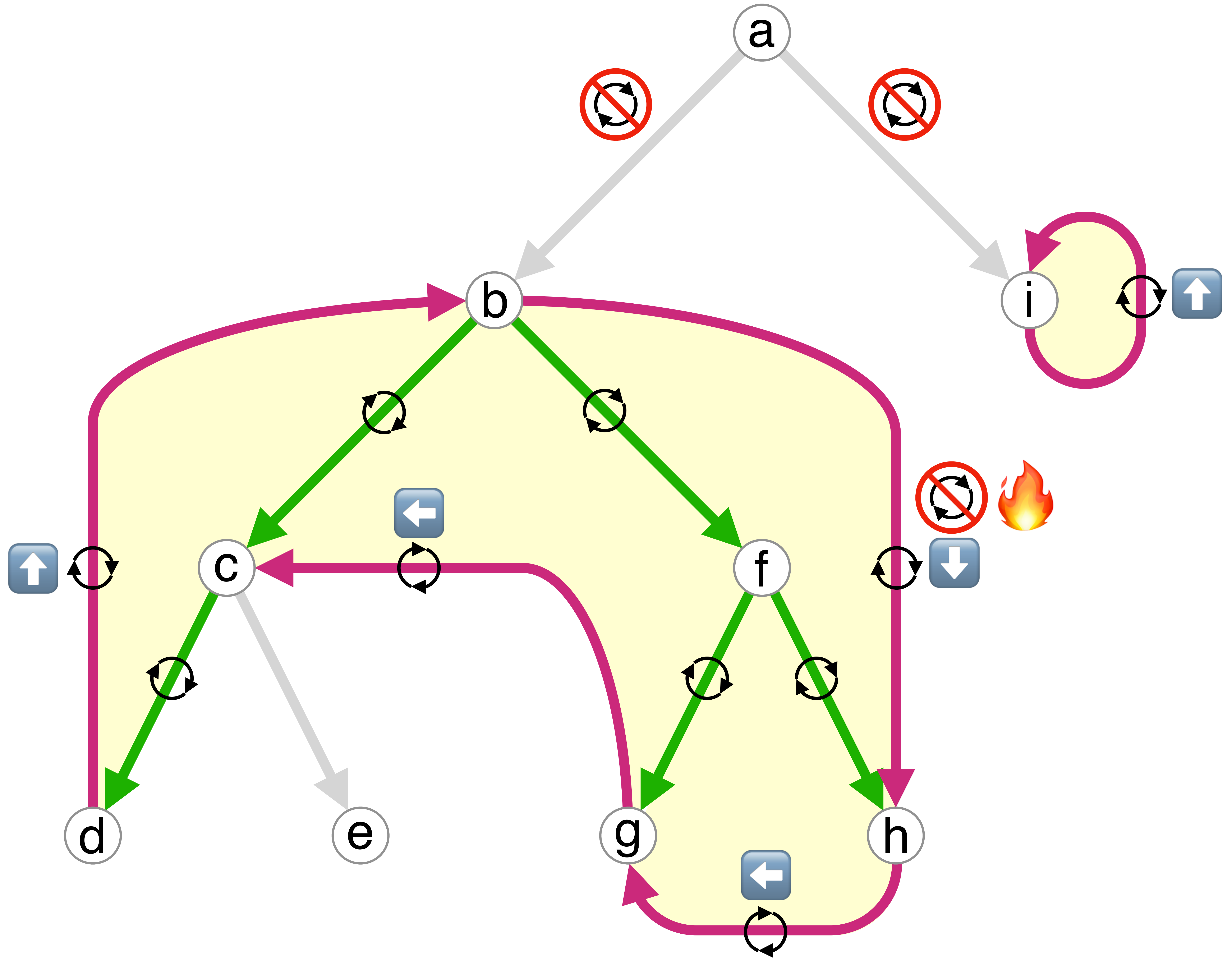
②




a
b
c
d
e
f
g
h
i

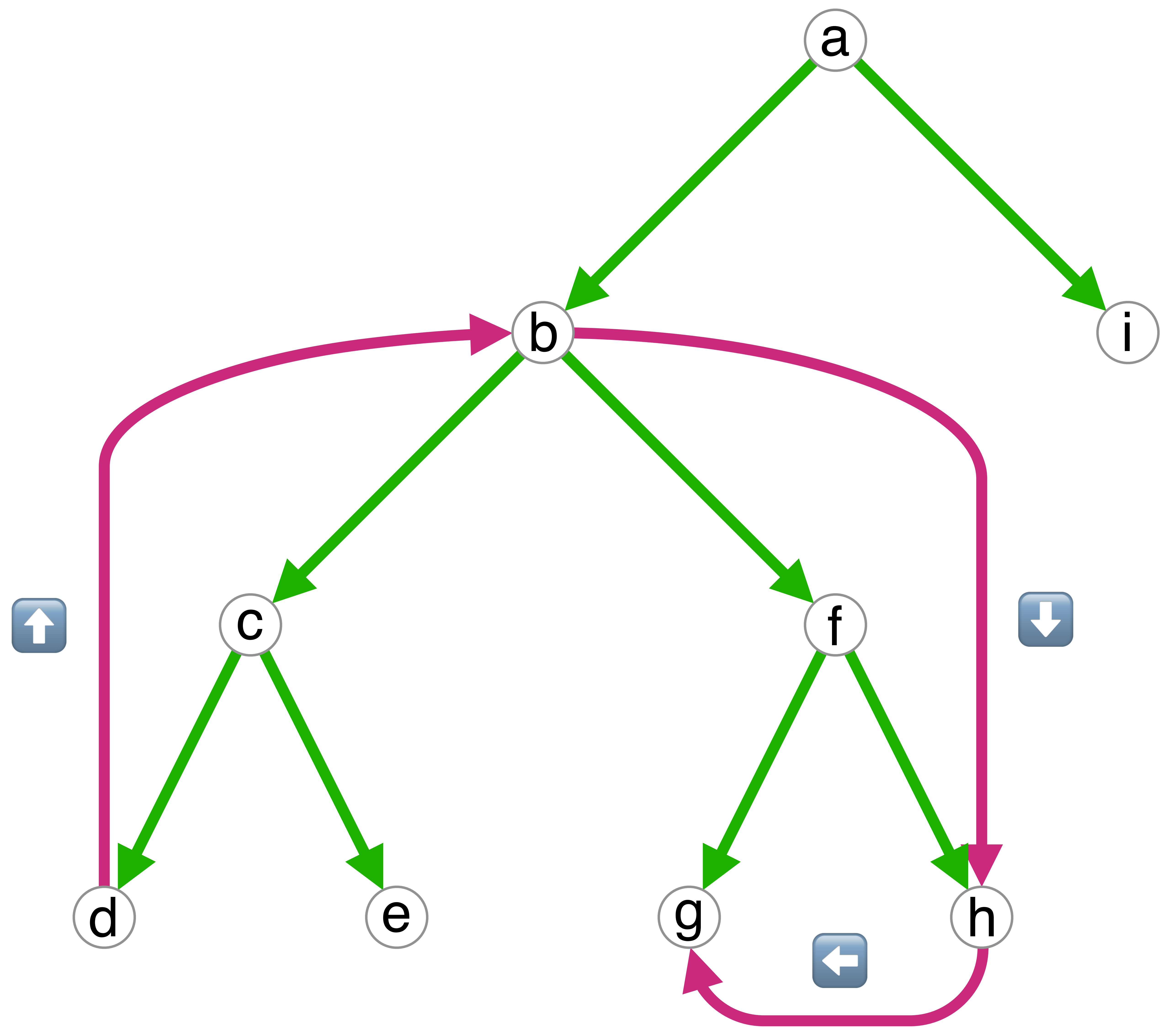

①
②
③


a
b
c
d
e
f
g
h
i

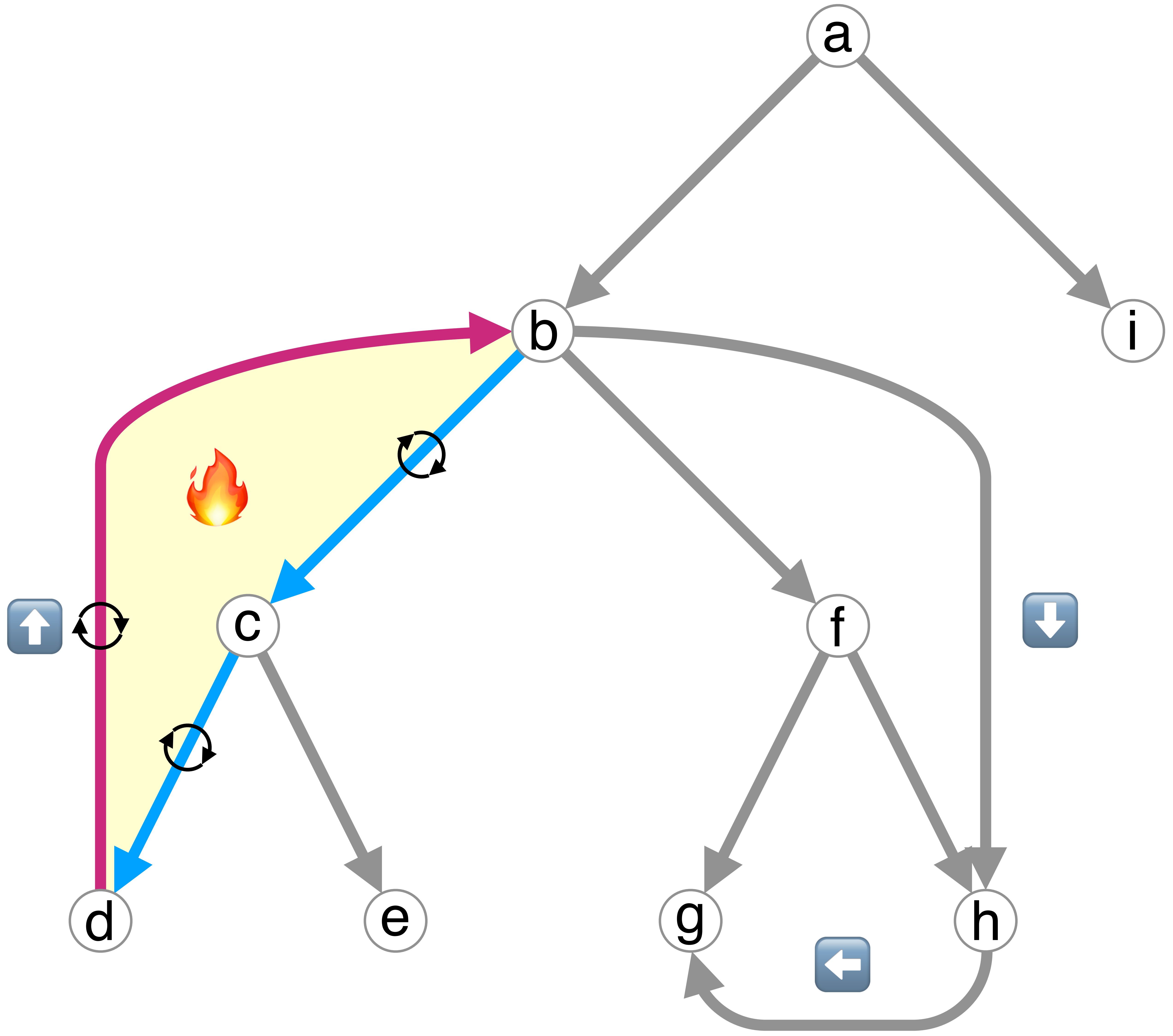


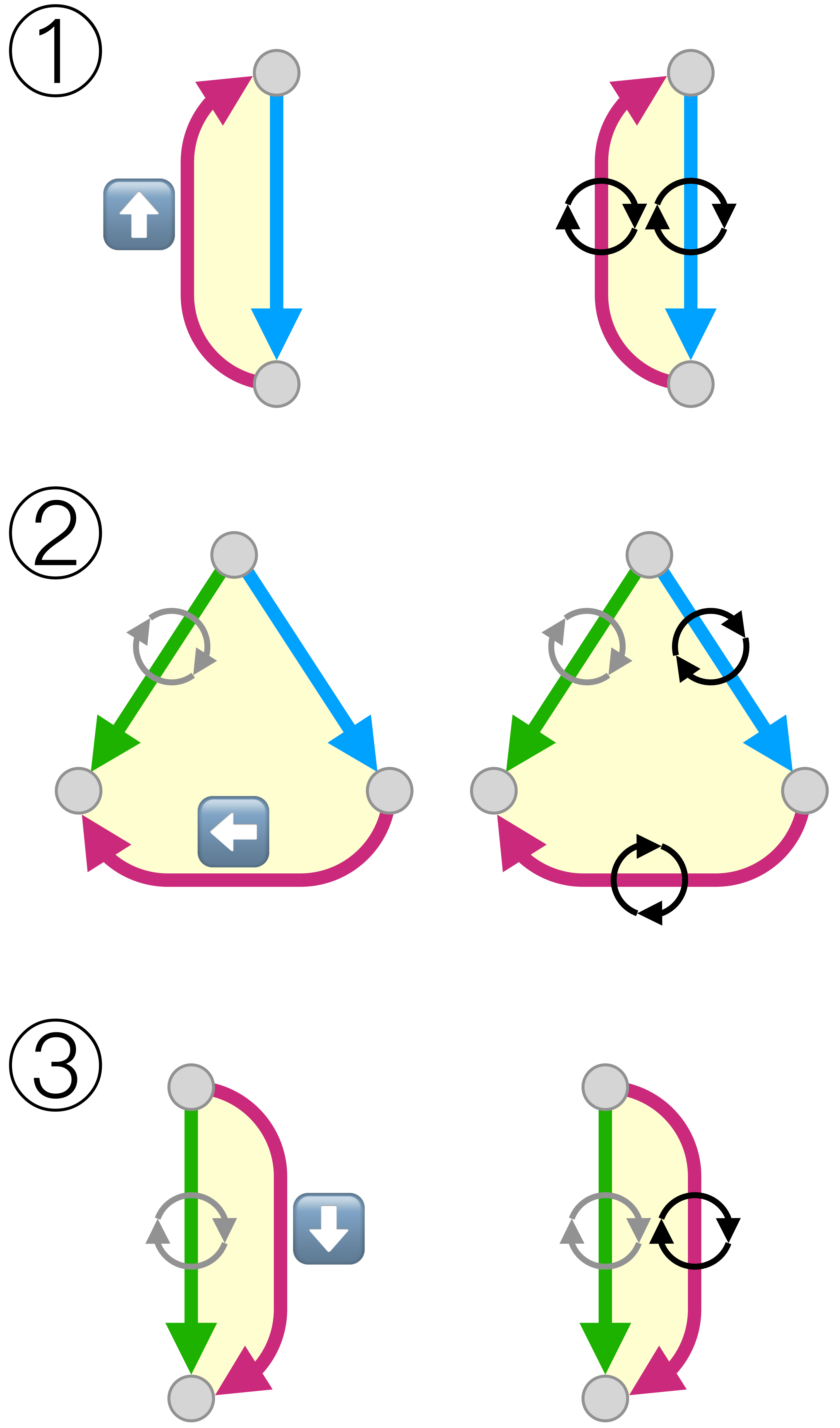
✅	$a \rightarrow b$	🚫		
✅	$a \rightarrow i$	🚫		
✅	$b \rightarrow c$		🔄	
✅	$b \rightarrow f$		🔄	
⬇️	$b \rightarrow h$	🚫	🔄	🔥
✅	$c \rightarrow d$		🔄	
✅	$c \rightarrow e$			
⬆️	$d \rightarrow b$		🔄	
✅	$f \rightarrow g$		🔄	
✅	$f \rightarrow h$		🔄	
⬅️	$g \rightarrow c$	✓	🔄	
⬅️	$h \rightarrow g$		🔄	
⬆️	$i \rightarrow i$	✓	🔄	

$a \rightarrow b$	⊘
$a \rightarrow i$	⊘
$b \rightarrow c$	
$b \rightarrow f$	
$b \rightarrow h$	⊘
$c \rightarrow d$	
$c \rightarrow e$	
$d \rightarrow b$	
$f \rightarrow g$	
$f \rightarrow h$	
$g \rightarrow c$	✓
$h \rightarrow g$	
$i \rightarrow i$	✓



- ✅ a → b 🚫
- ✅ a → i 🚫
- ✅ b → c
- ✅ b → f
- ⬇️ b → h 🚫
- ✅ c → d
- ✅ c → e
- ⬆️ d → b
- ✅ f → g
- ✅ f → h
- 🔄
- ⬅️ h → g
- 🔄


a
b
c
d
e
f
g
h
i


①
②
③

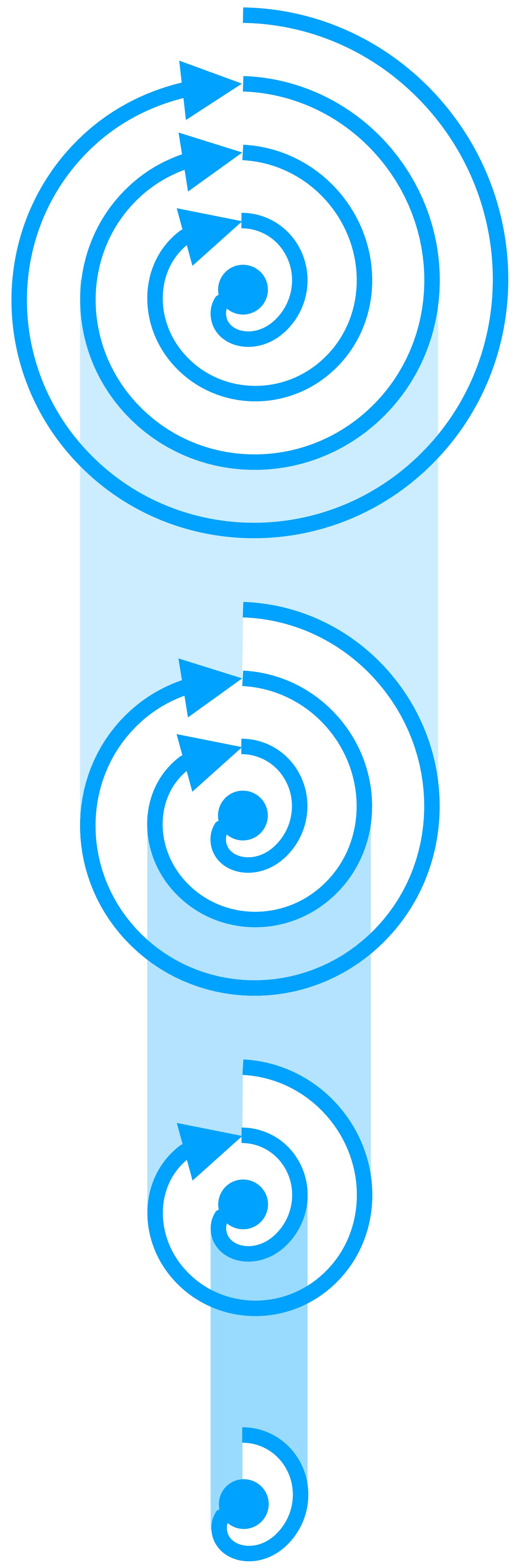
```
unsigned count_recursively( char *b, char *e )
interface
  {
   claim recursively_reachable( b, e );
   implementation;
   claim result == std::count( b, e );
   claim usable( result );
  }
implementation
  {
   if ( b == e )
      return 0u;
   else
      return count_recursively( ++b, e ) + 1u;
  }
```

```
inline claimable
recursively_reachable( char *b, char *e )
  {
   if ( b != e )
      entail recursively_reachable( ++b, e );
  }
```

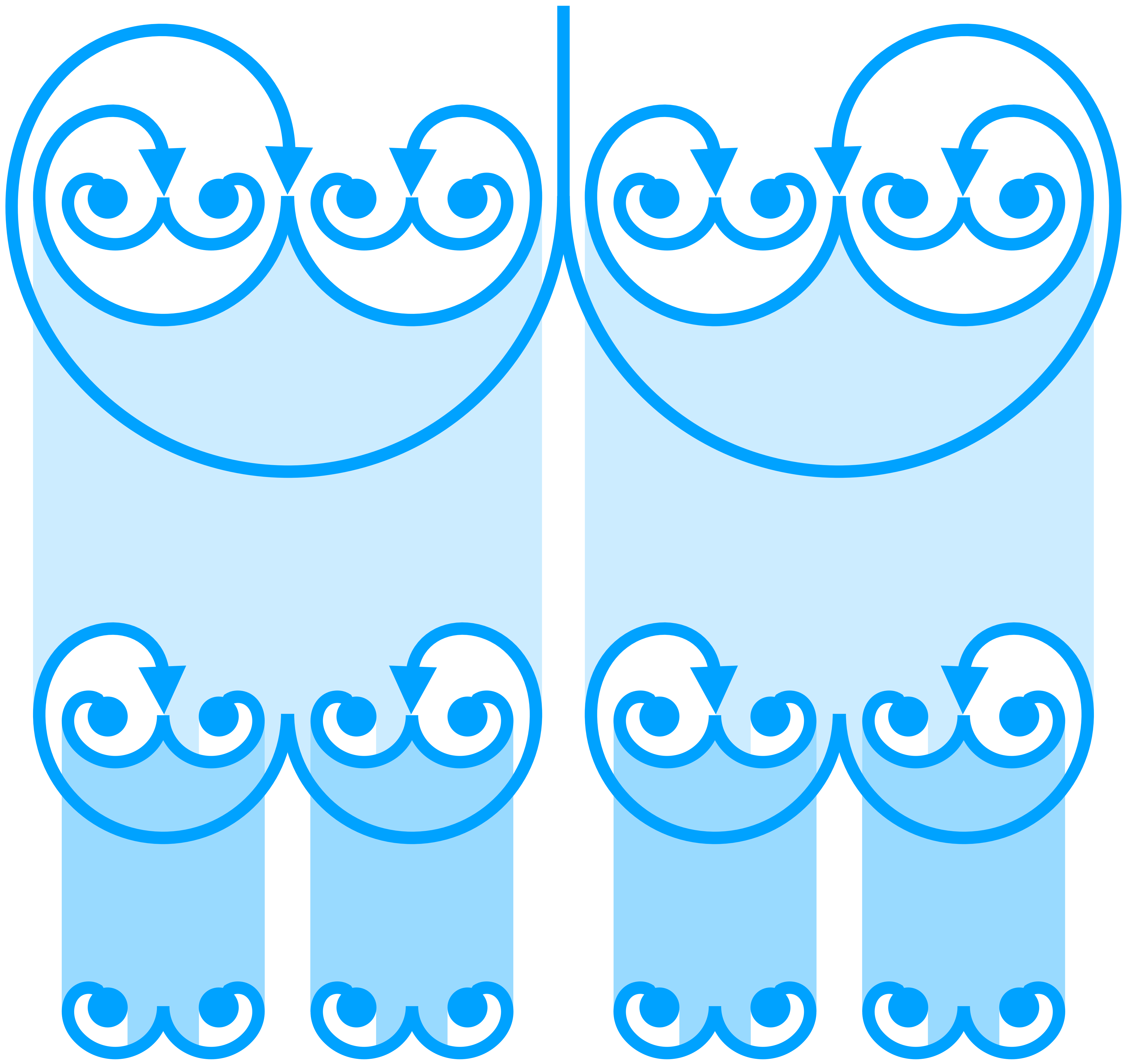
In my earlier talks, “**entail**” was “**require**.”

```
inline claimable
recursively_reachable( char *b, char *e )
  {
   if ( b != e )
       entail recursively_reachable( ++b, e );
  }
```

```
inline claimable
recursively_reachable( char *b, char *e )
  {
   if ( b != e )
      entail recursively_reachable( ++b, e );
  }
```

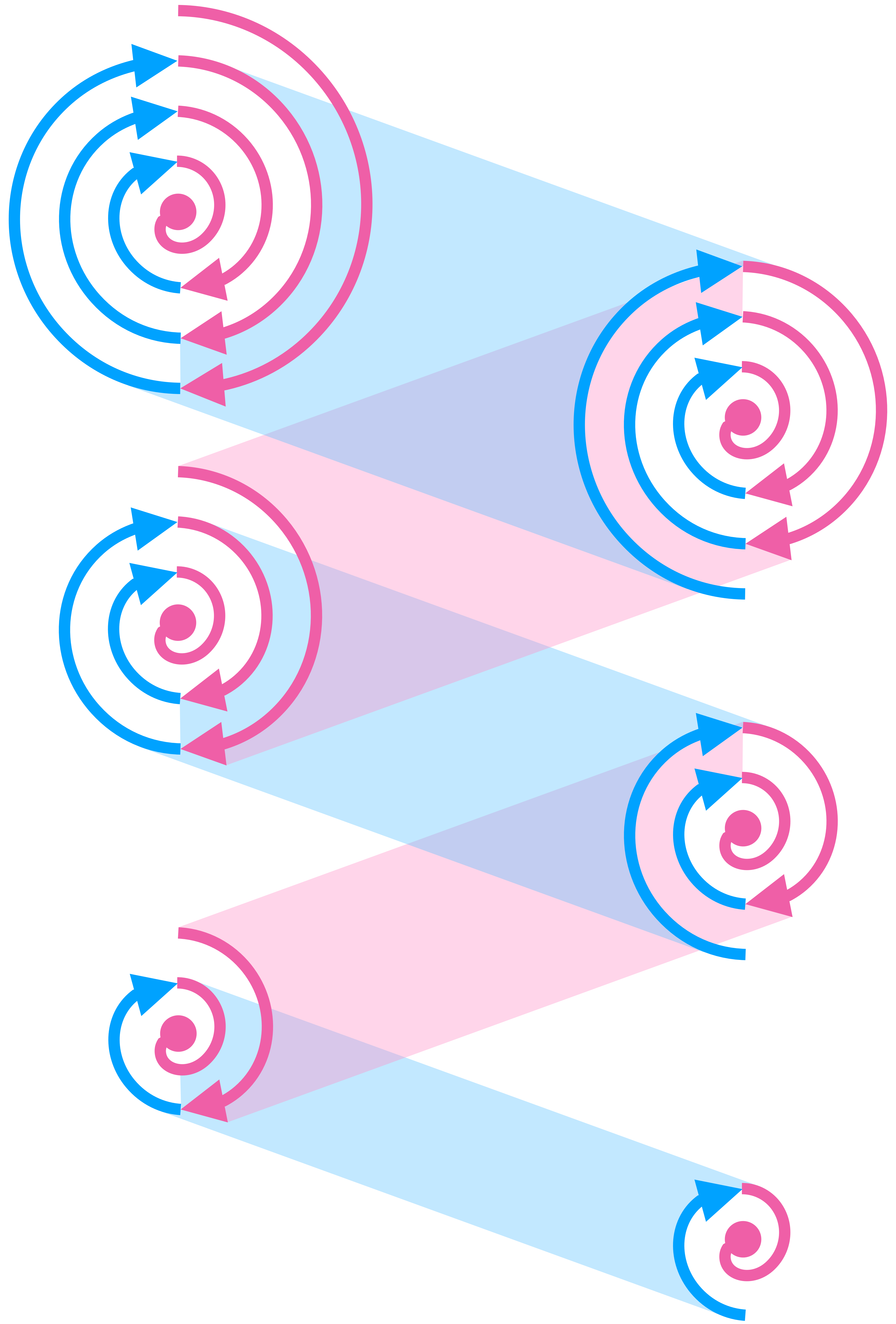


```
inline claimable
recursively_traversable( tree& t )
  {
   if ( t.left() != nullptr )
      entail recursively_traversable( *t.left() );

   if ( t.right() != nullptr )
      entail recursively_traversable( *t.right() );
  }
```



```
inline claimable
recursively_up1_down2( int n )
   {
    if ( n > 0 )
        entail recursively_down2_up1( n+1 );
   }
```

```
inline claimable
recursively_down2_up1( int n )
   {
    if ( n >= 2 )
        entail recursively_up1_down2( n-2 );
   }
```



Represent like with like.

“Stat rosa pristina nomine,
nomina nuda tenemus.”

Thank you for listening.

Questions?