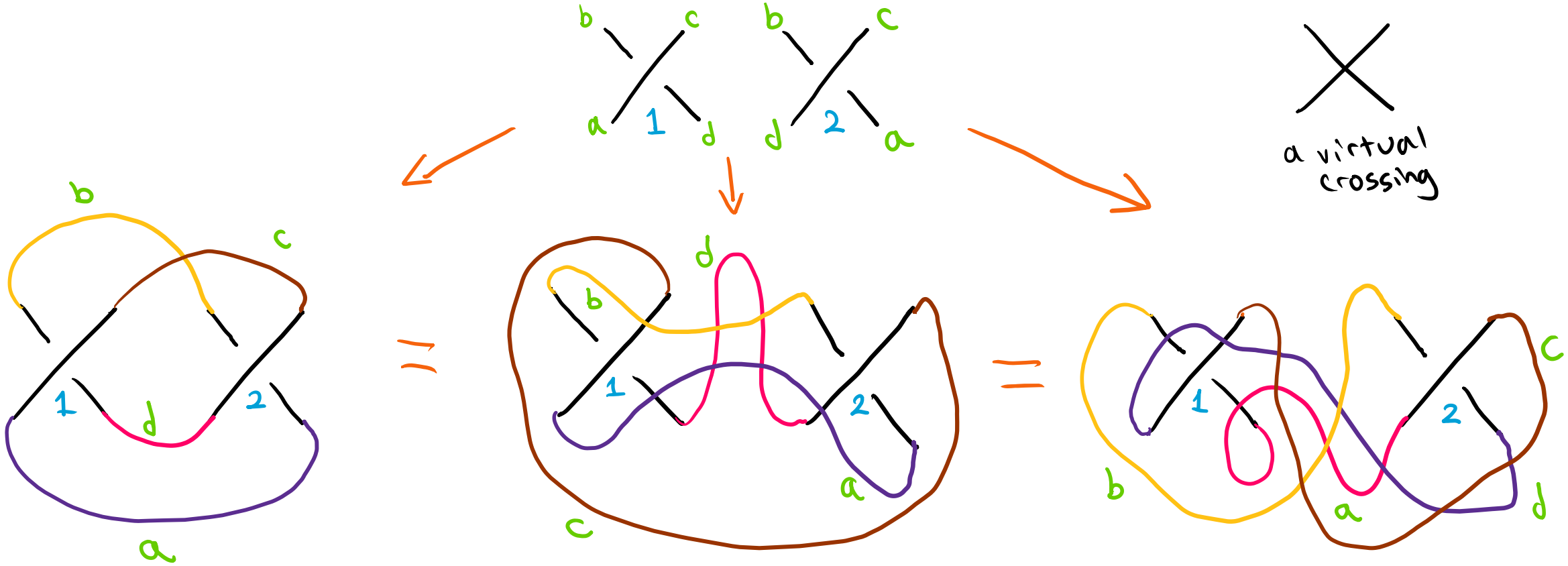
A Proof that Classical Knots Embed in Virtual Knots

Recap: Virtual Knot Diagrams

Virtual knot diagrams are not really diagrams, but rather valid "**crossing structures**". Any virtual knot diagram does however admit a non-unique **drawing**, which in general makes use of **virtual crossings**.

Given a drawing of a virtual knot diagram, you can move segments around **virtually** to obtain another drawing.

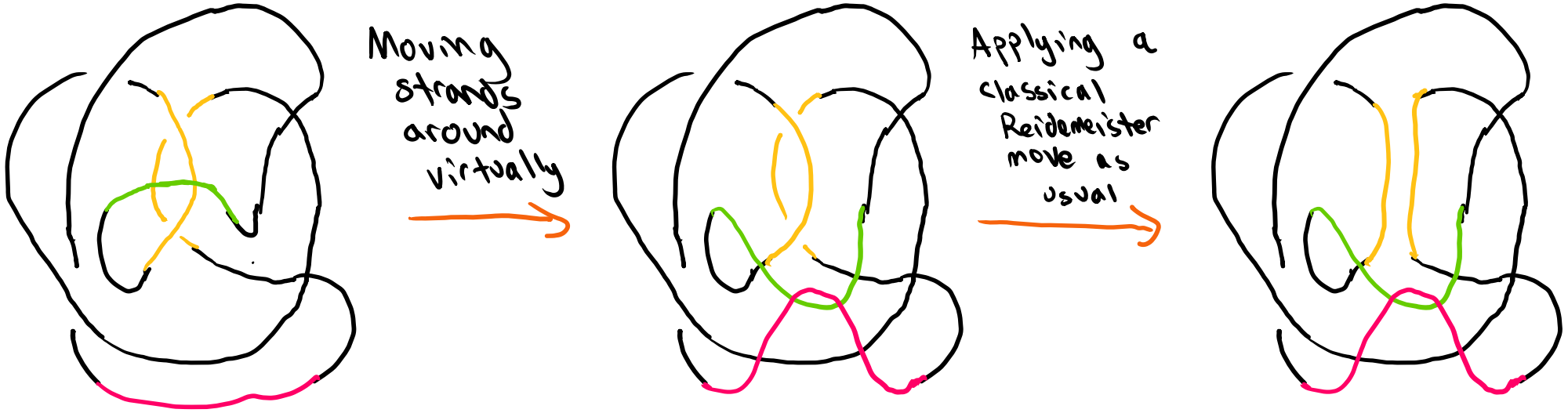


Recap: Virtual Reidemeister Moves

Virtual Reidemeister moves are classical Reidemeister moves, but with the non-uniqueness of the drawing of a virtual knot diagram taken into account.

In terms of drawings, the application of a virtual Reidemeister move consists of the following.

1. Moving strands of the diagram around virtually.
2. Applying a classical Reidemeister move as usual.



Recap: The Main Result

Classical knot diagrams are virtual knot diagrams, and classical Reidemeister moves are virtual Reidemeister moves.

Therefore, two classical knot diagrams equivalent as classical knot diagrams are equivalent as virtual knot diagrams.

The result we prove is the converse.

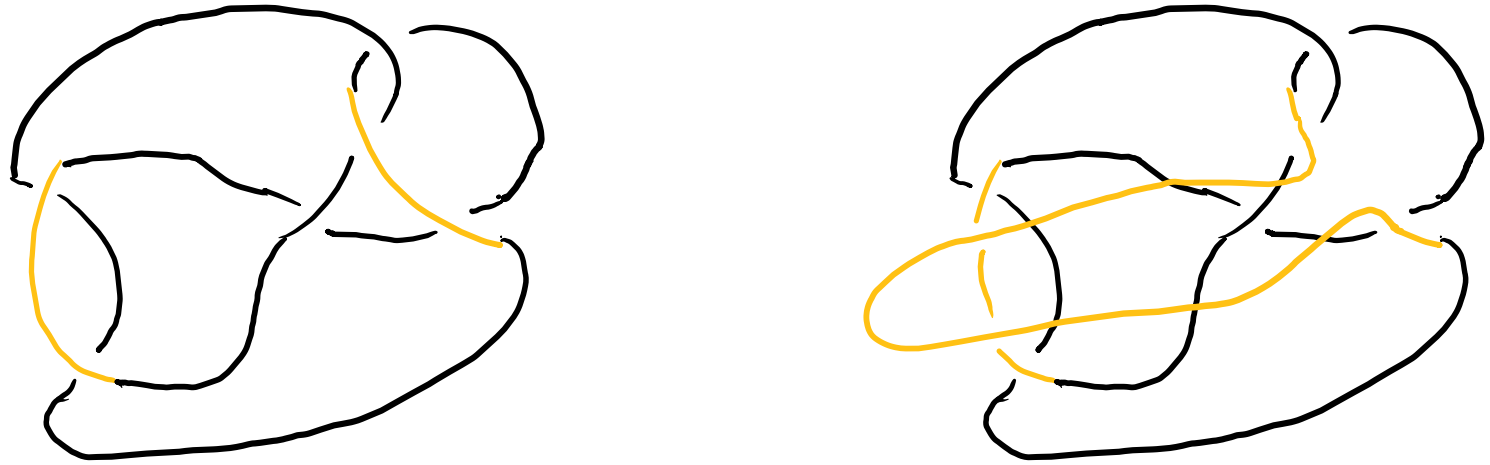
Theorem. *If two classical knot diagrams are equivalent as virtual knot diagrams, then they are equivalent as classical knot diagrams.*

Recap: Local vs. Non-Local Moves

One can consider classical Reidemeister moves from the **virtual perspective**. That is, one can consider how classical Reidemeister moves impact the crossing structure of diagrams. Doing so leads to a classification of virtual Reidemeister moves.

Local virtual Reidemeister moves are ones that "look classical" when considering their impact on the crossing structure.

Non-local virtual Reidemeister moves are ones with a **fundamentally non-classical** impact on the crossing structure.



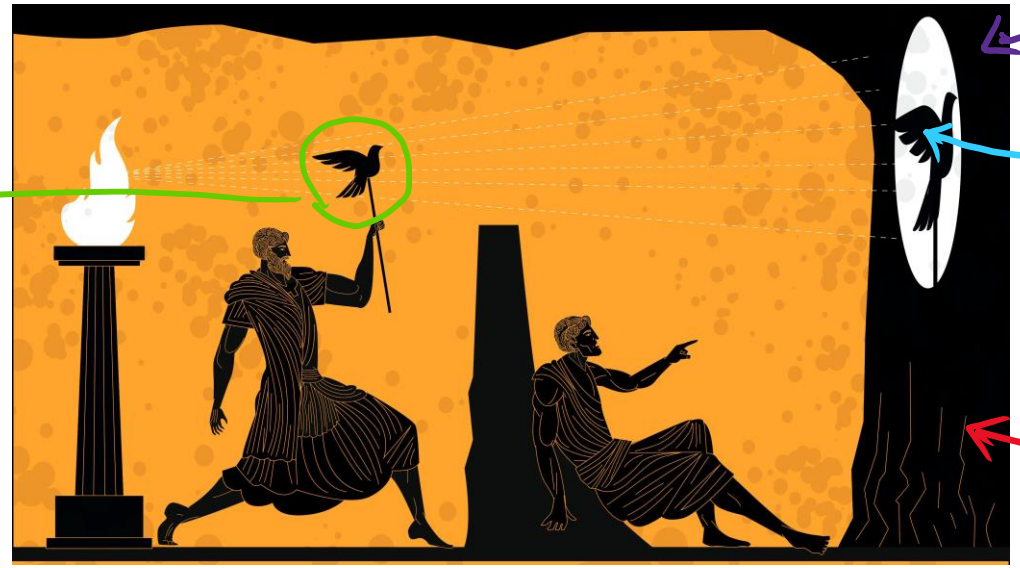
Example of a non-local move

Recap: Local vs. Non-Local Moves

One can consider classical Reidemeister moves from the **virtual perspective**. That is, one can consider how classical Reidemeister moves impact the crossing structure of diagrams. Doing so leads to a classification of virtual Reidemeister moves.

Local virtual Reidemeister moves are ones that "look classical" when considering their impact on the crossing structure.

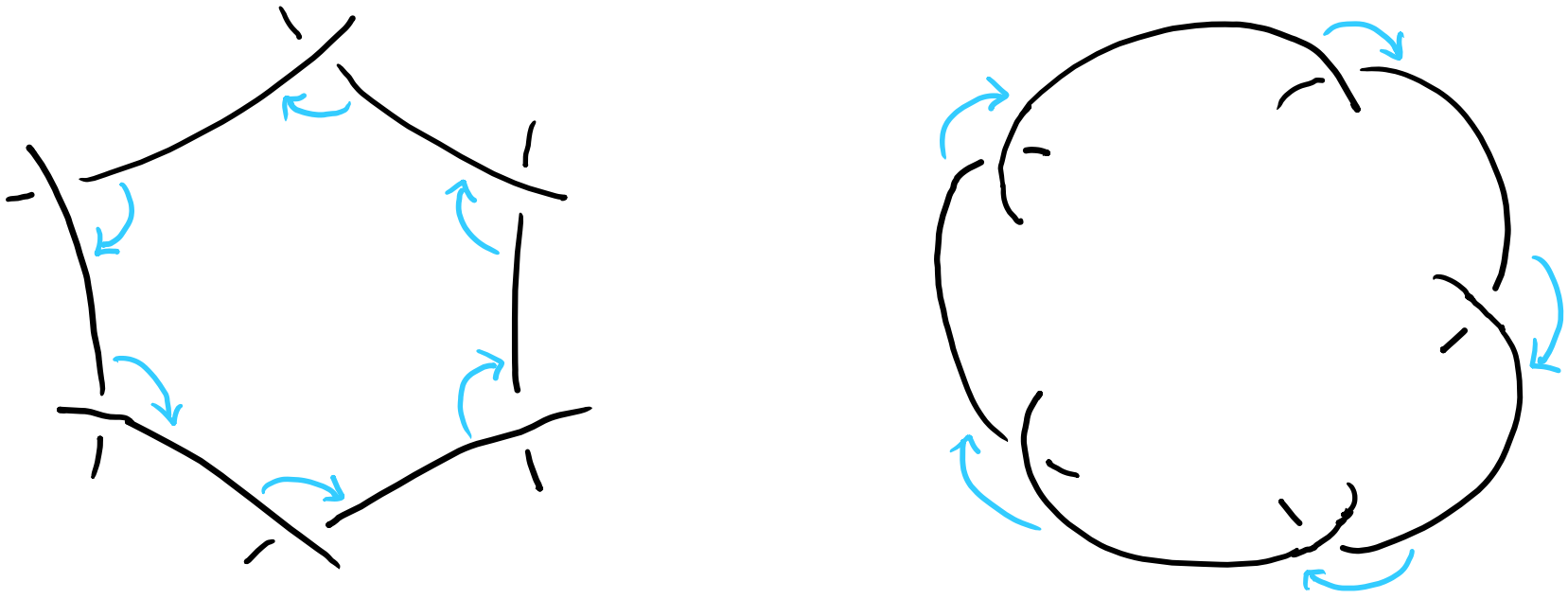
Non-local virtual Reidemeister moves are ones with a **fundamentally non-classical** impact on the crossing structure.



Recap: Bounding Cycles

Bounding cycles are the virtual analogue of the cycles of a classical knot diagram which bound **faces** of the diagram.

They admit a combinatorial characterization: a bounding cycle can be ordered so that each edge can be obtained from the previous edge by rotating clockwise at a common crossing.

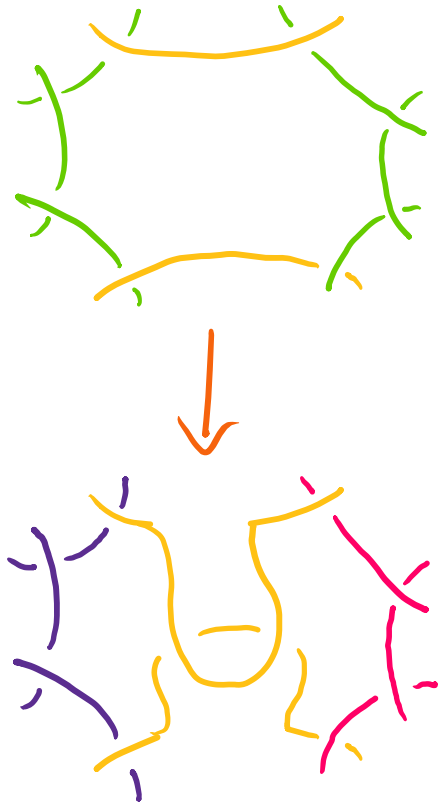


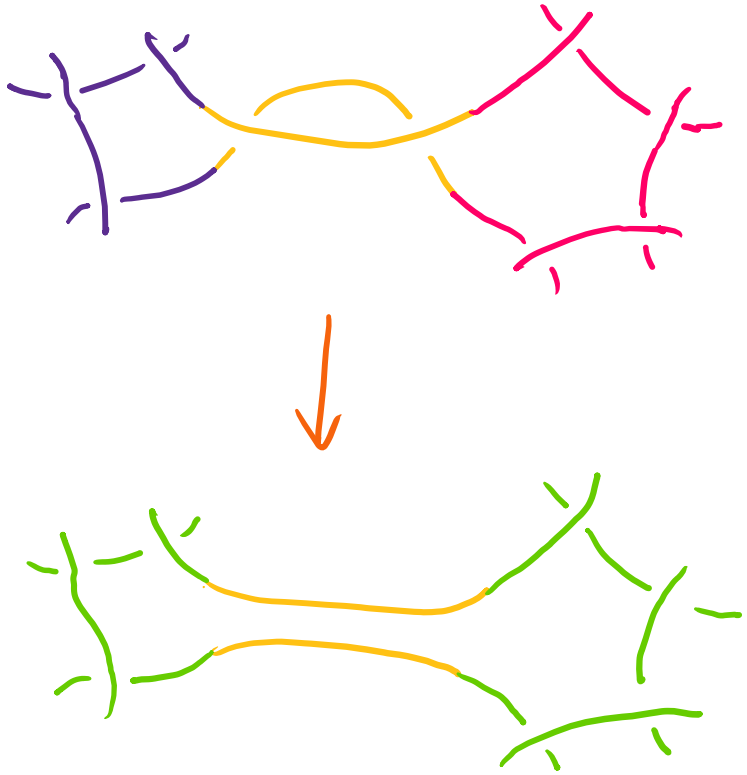
Examples of bounding cycles

Recap: Local vs. Non-Local Moves

We consider all virtual Reidemeister-I and virtual Reidemeister-III moves to be **local**.

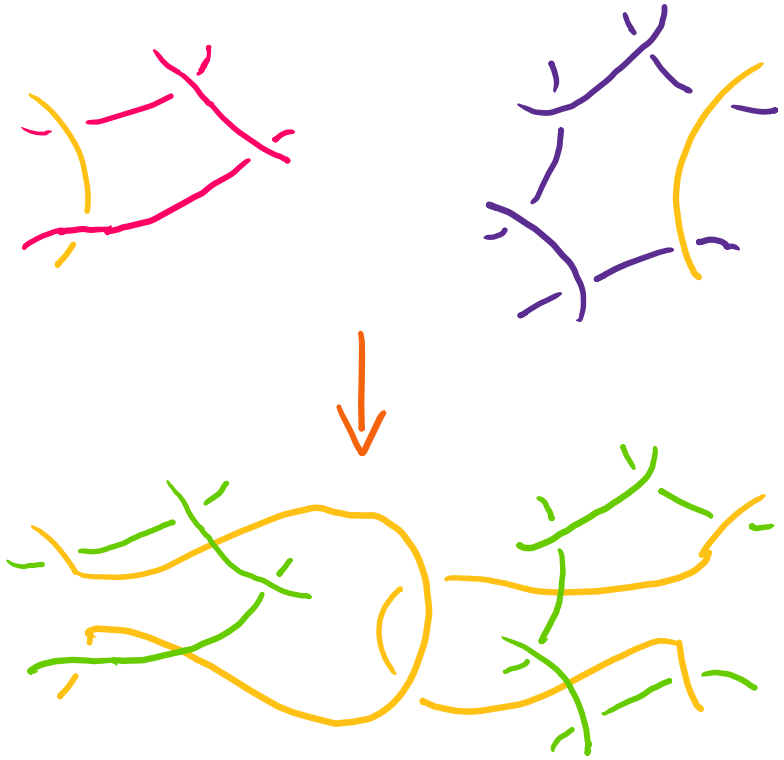
We consider the following kinds of virtual Reidemeister-II moves to be **local** as well.



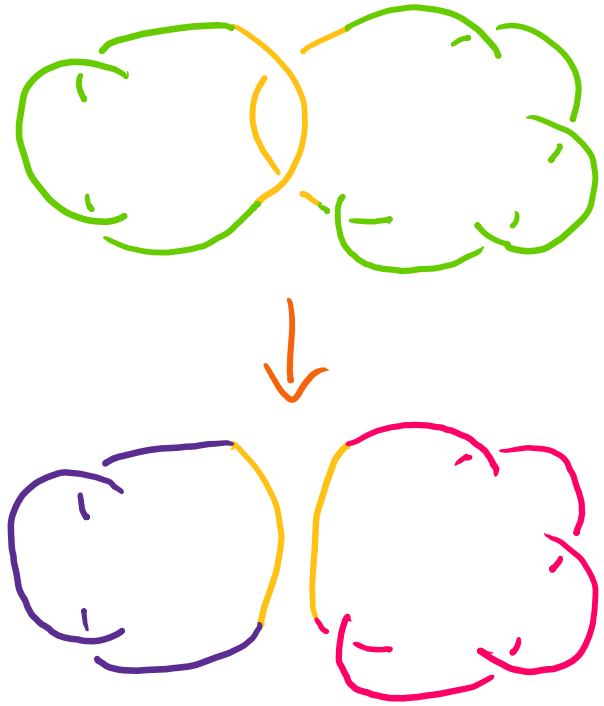


Recap: Local vs. Non-Local Moves

The only moves we consider to be **non-local** are the following kinds of virtual Reidemeister-II moves.



This move is impossible to perform classically.



The structure on the top could never appear classically.

Reformulation of the Main Result

Using the concept of local/non-local moves, we can eliminate classical Reidemeister moves from the statement of the main result.

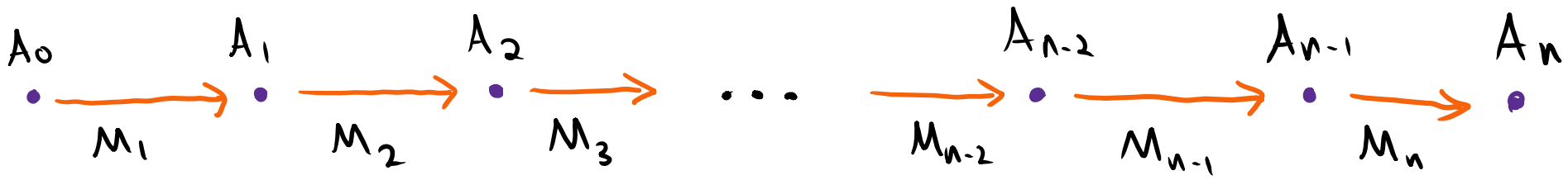
We prove the following equivalent statement instead.

Theorem. *If two classical knot diagrams can be related via a sequence of virtual Reidemeister moves, then they can be related by a sequence of only local virtual Reidemeister moves.*

Notation

$A \rightarrow_M A'$: the virtual knot diagram A' is obtained from A via an application of the move M.

We define a **path** to be a sequence of virtual knot diagrams $(A_0, \ldots, A_n)$ such that $A_0 \rightarrow_{M_1} \ldots \rightarrow_{M_n} A_n$.



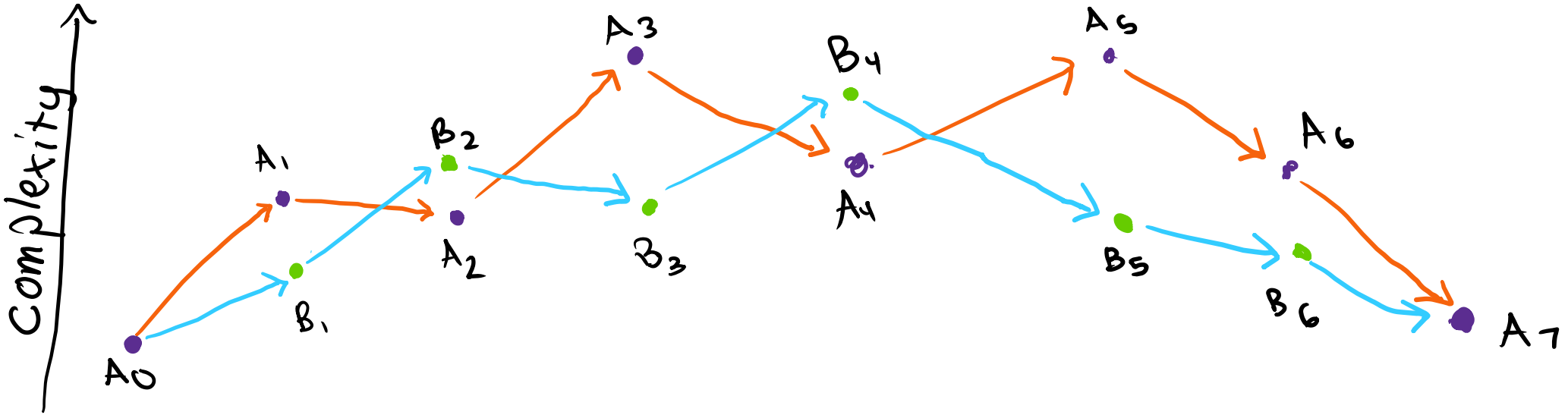
Proof Setup and Preview

Suppose A and A' are classical knot diagrams connected by a path $A_0 \to_{M_1} \ldots \to_{M_n} A_n$, where $A_0 = A$ and $A_n = A'$, and suppose this path contains non-local moves.

We naturally consider diagrams with more crossings to be **more complicated** than diagrams with less crossings.

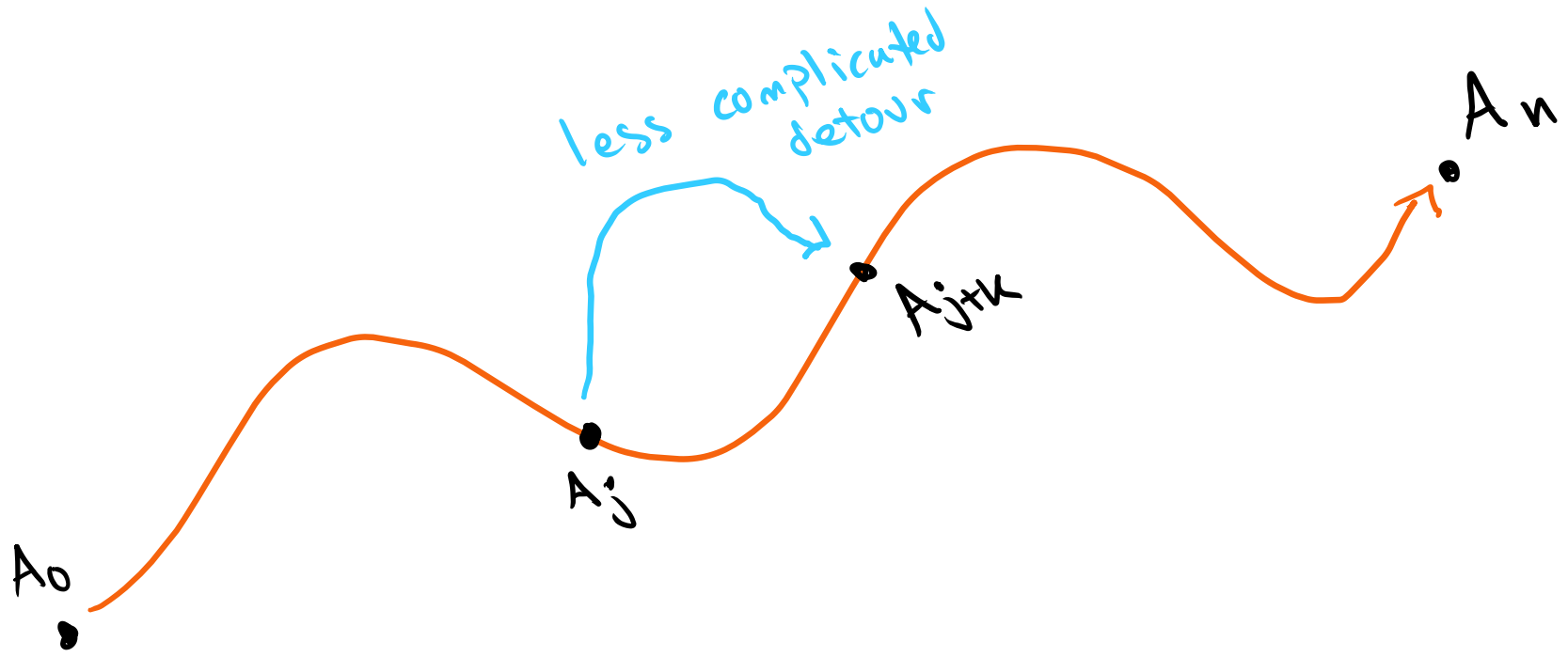
We will show that since our path uses non-local moves, it is not optimal in the sense that there is a less complicated* path between A and A'.

Repeating the argument, it follows that there is a path between A and A' using only local moves, as desired.



Strategy

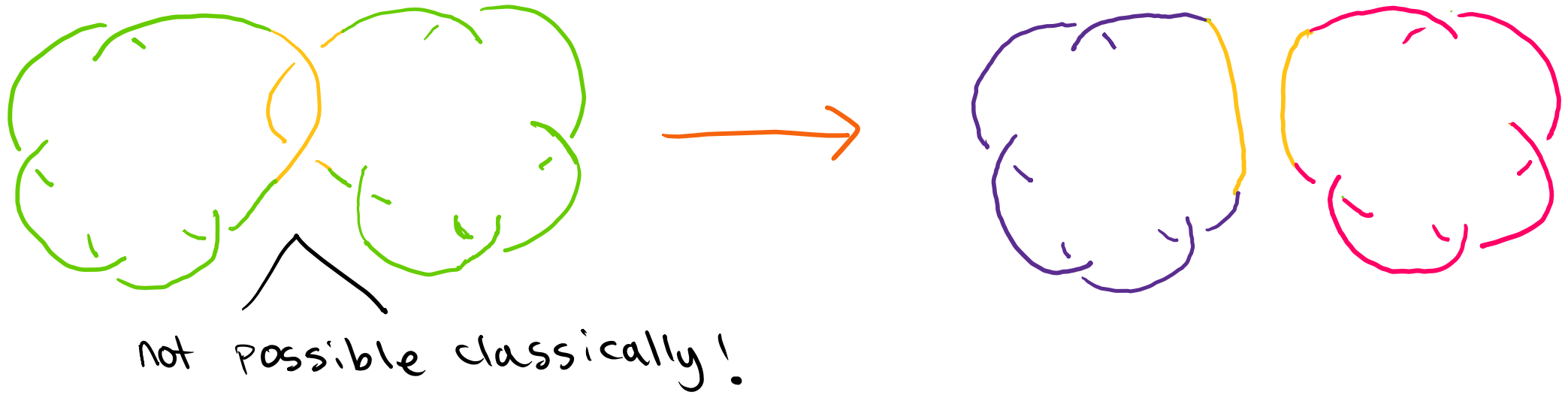
Our strategy will be to find a specific kind of subpath, which we call a **plateau**, and find a **detour** which is less complicated than this subpath.



Focusing on a Plateau

Consider the non-local moves applied in our path.

A non-local backwards VR-II move cannot be applied to a classical knot diagram, so the first non-local move must be forwards.

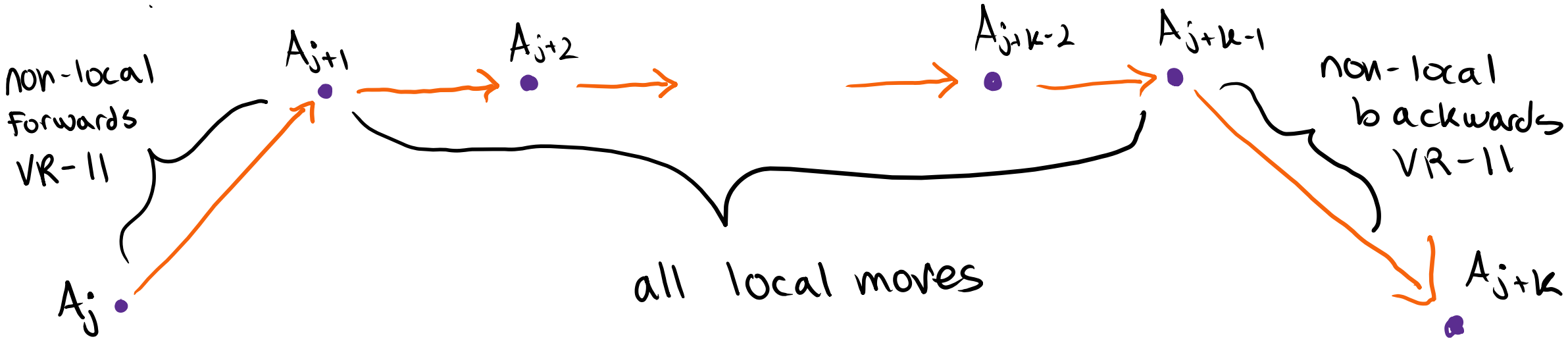


By symmetry, the last non-local move must be backwards.

Focusing on a Plateau

The first non-local move must be forwards, and the last non-local move must be backwards.

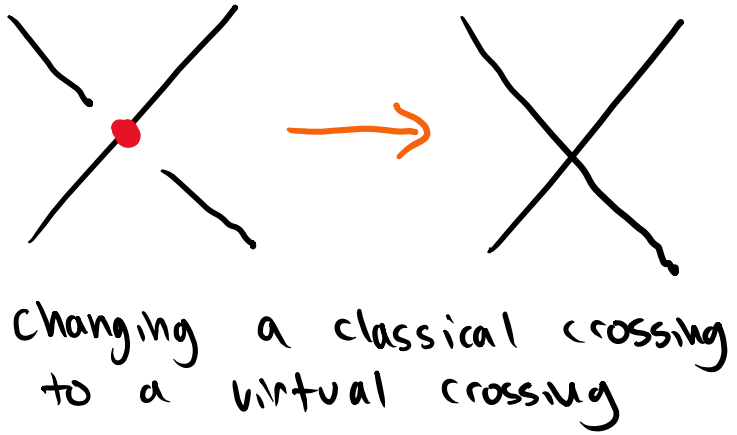
Therefore, we can find a subpath $A_j \rightarrow_{M_{j+1}} \ldots \rightarrow_{M_{j+k}} A_{j+k}$ of our original path $A_0 \rightarrow_{M_1} \ldots \rightarrow_{M_n} A_n$ which is a **plateau**, i.e., a subpath starting with a non-local forwards move, followed by a series of local moves, and ending with a non-local backwards move.



Removing Crossings

There is a natural way to remove crossings from a virtual knot diagram (and have the result be a virtual knot diagram too).

In terms of drawings, removal amounts to changing a classical crossing to a virtual crossing.



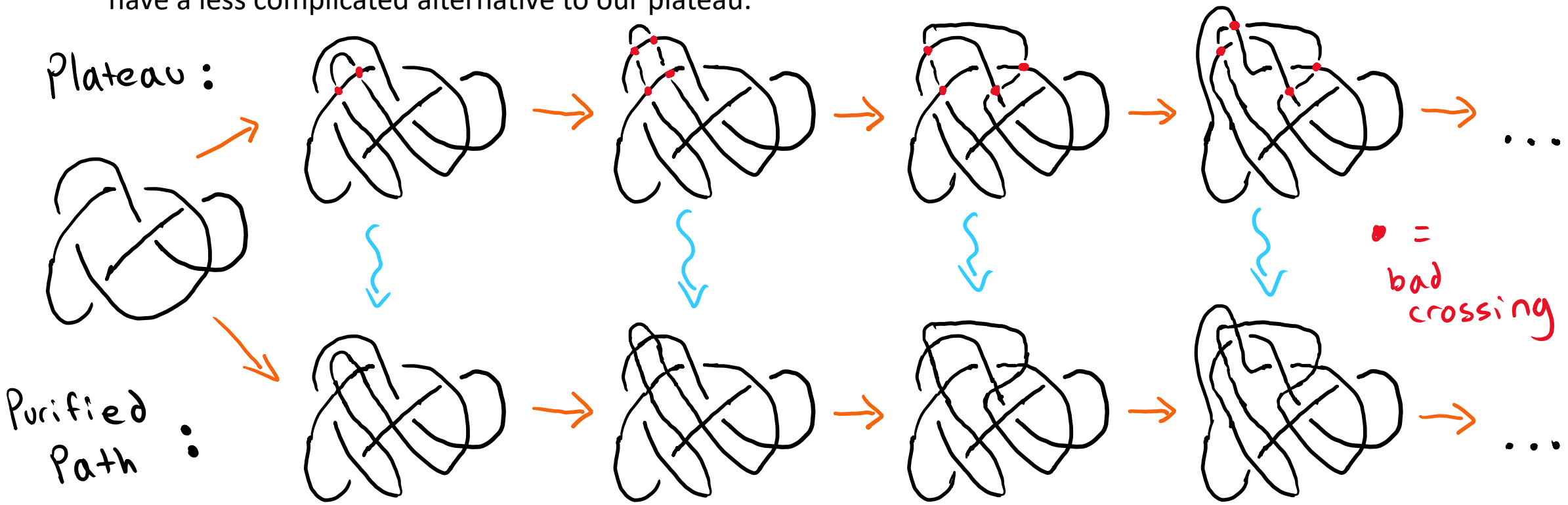
Notation: For a virtual knot diagram A and a set X of its crossings, we let $A - X$ denote the result of removing the crossings in X from A.

Strategy: Purify our Plateau

Our strategy will be to first declare each crossing of each diagram in our plateau to be either **good** or **bad**.

We will then **purify** the diagrams in our plateau by **removing all bad crossings**.

The goal is to choose bad crossings in a manner so that the purified diagrams form a path of their own, so we have a less complicated alternative to our plateau.



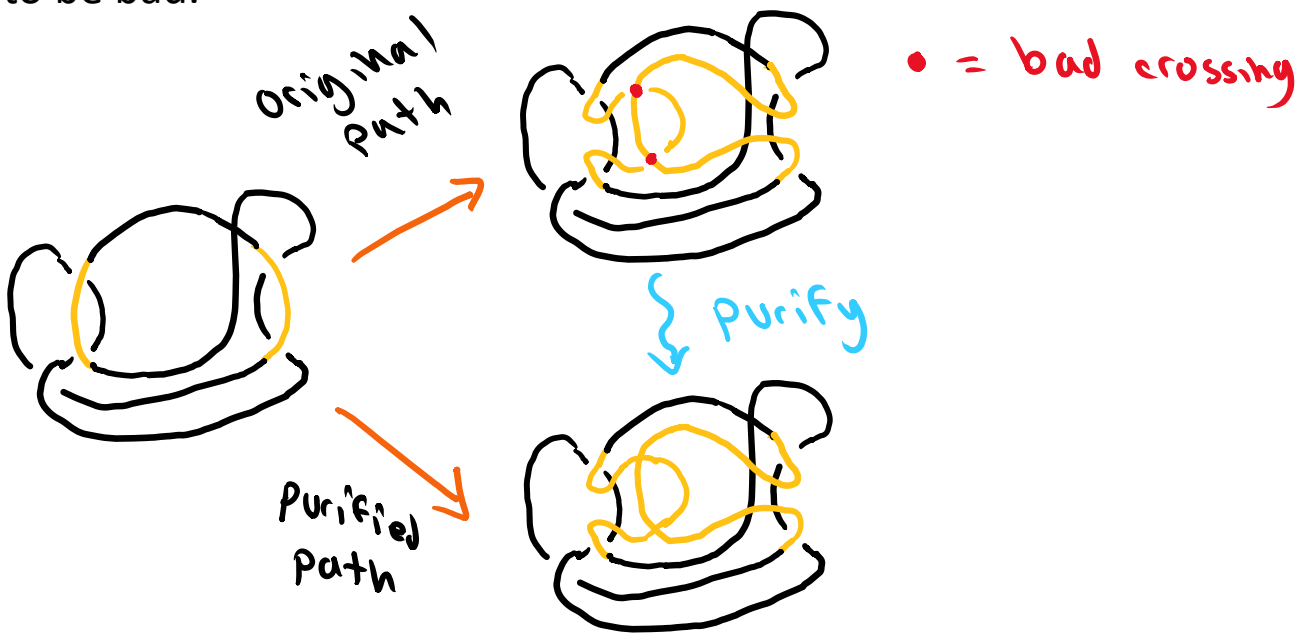
Judging Crossings

To make our strategy work, we have to find a method for judging crossings to be good or bad.

Our method will be **inductive.**

We declare all crossings of the first diagram A_j to be good, because we want our purified path to start at A_j as well.

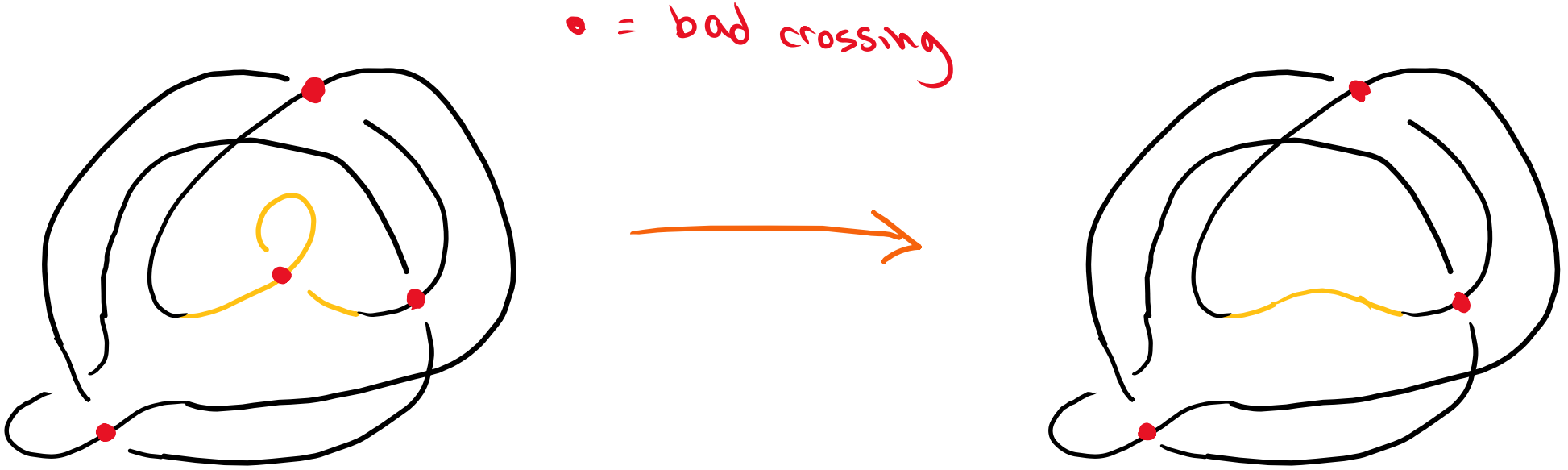
Naturally, we declare the two crossings introduced by the first move M_{j+1} in our plateau, which is a non-local forwards VR-II move, to be bad.



Judgment Rules

Inductively, we judge crossings to be good or bad after applying a virtual Reidemeister move according to the following rules.

1. Crossings which are not destroyed by the move maintain their goodness or badness.
2. Crossings created by the move must all be judged the same (either all good or all bad).

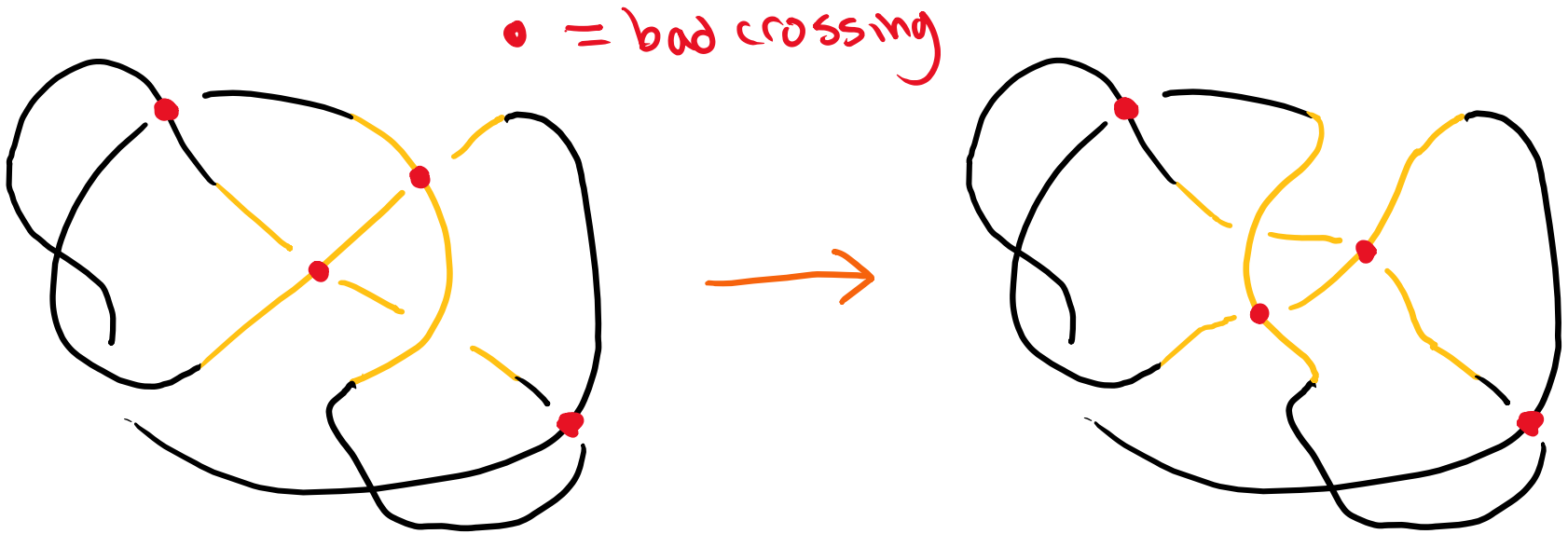


Rule 1 example

Judgment Rules

Inductively, we judge crossings to be good or bad after applying a virtual Reidemeister move according to the following rules.

1. Crossings which are not destroyed by the move maintain their goodness or badness.
2. Crossings created by the move must all be judged the same (either all good or all bad).

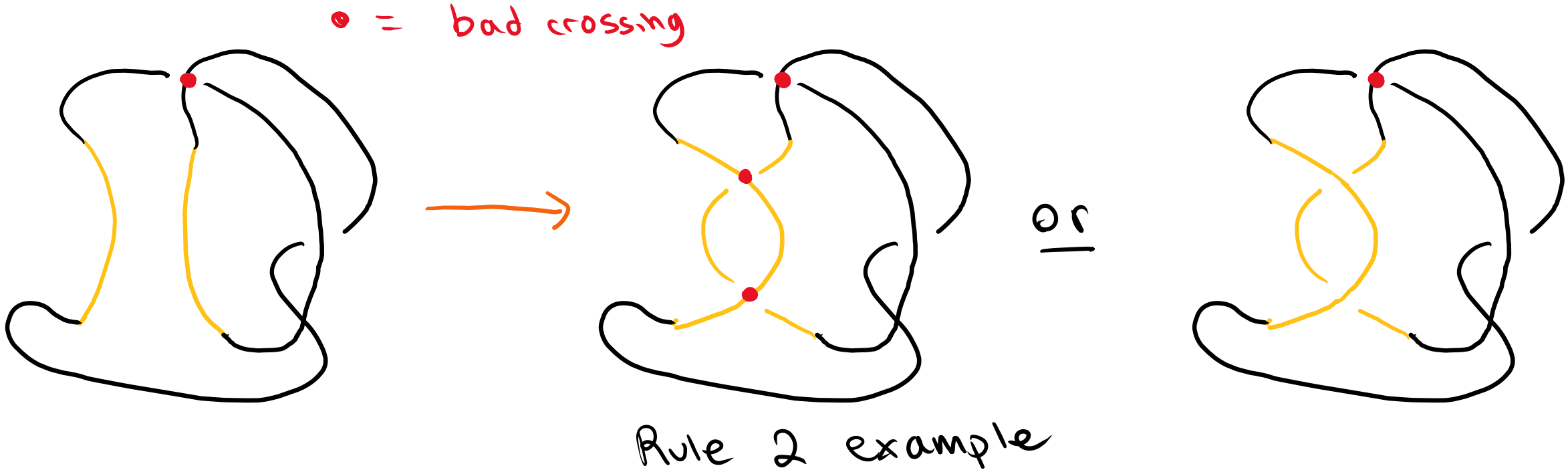


Rule 1 example

Judgment Rules

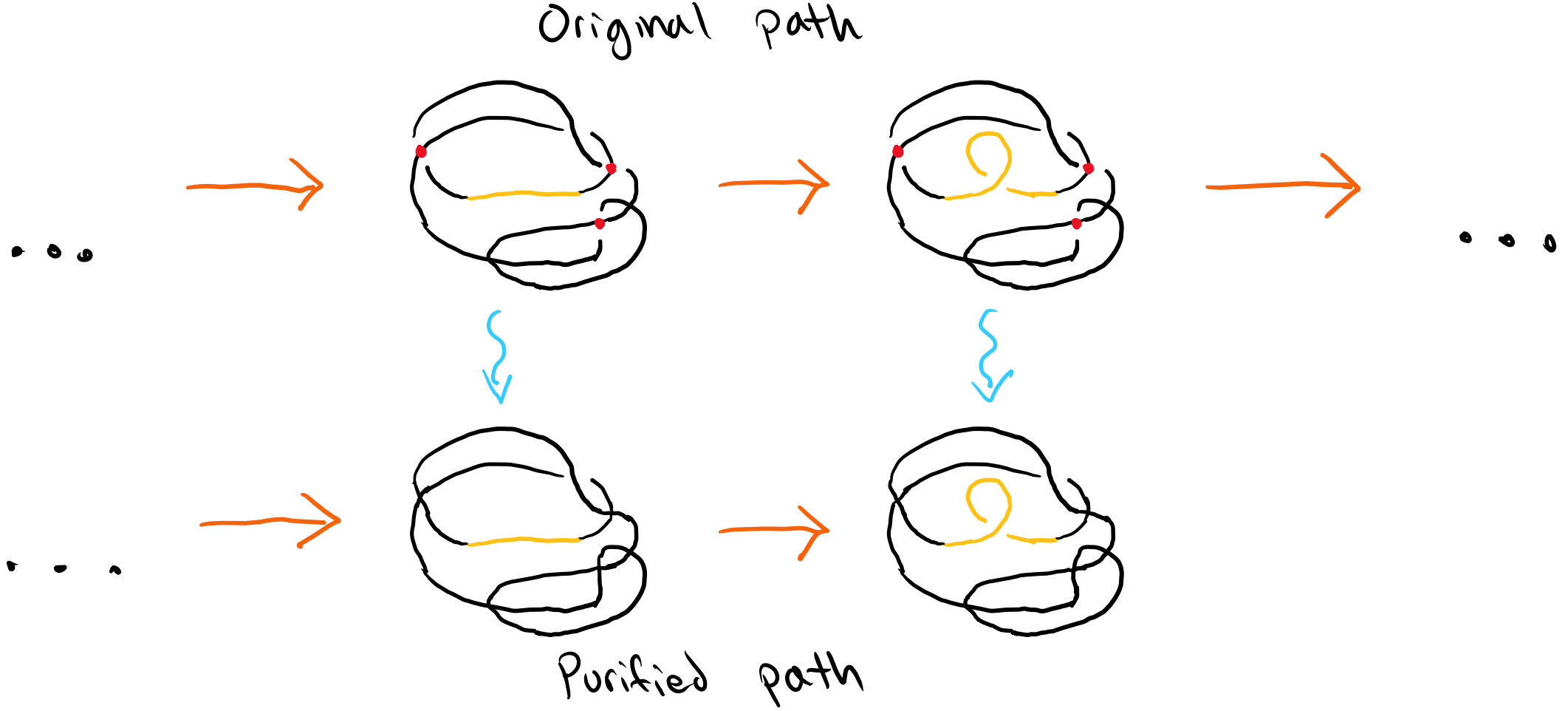
Inductively, we judge crossings to be good or bad after applying a virtual Reidemeister move according to the following rules.

1. Crossings which are not destroyed by the move maintain their goodness or badness.
2. Crossings created by the move must all be judged the same (either all good or all bad).



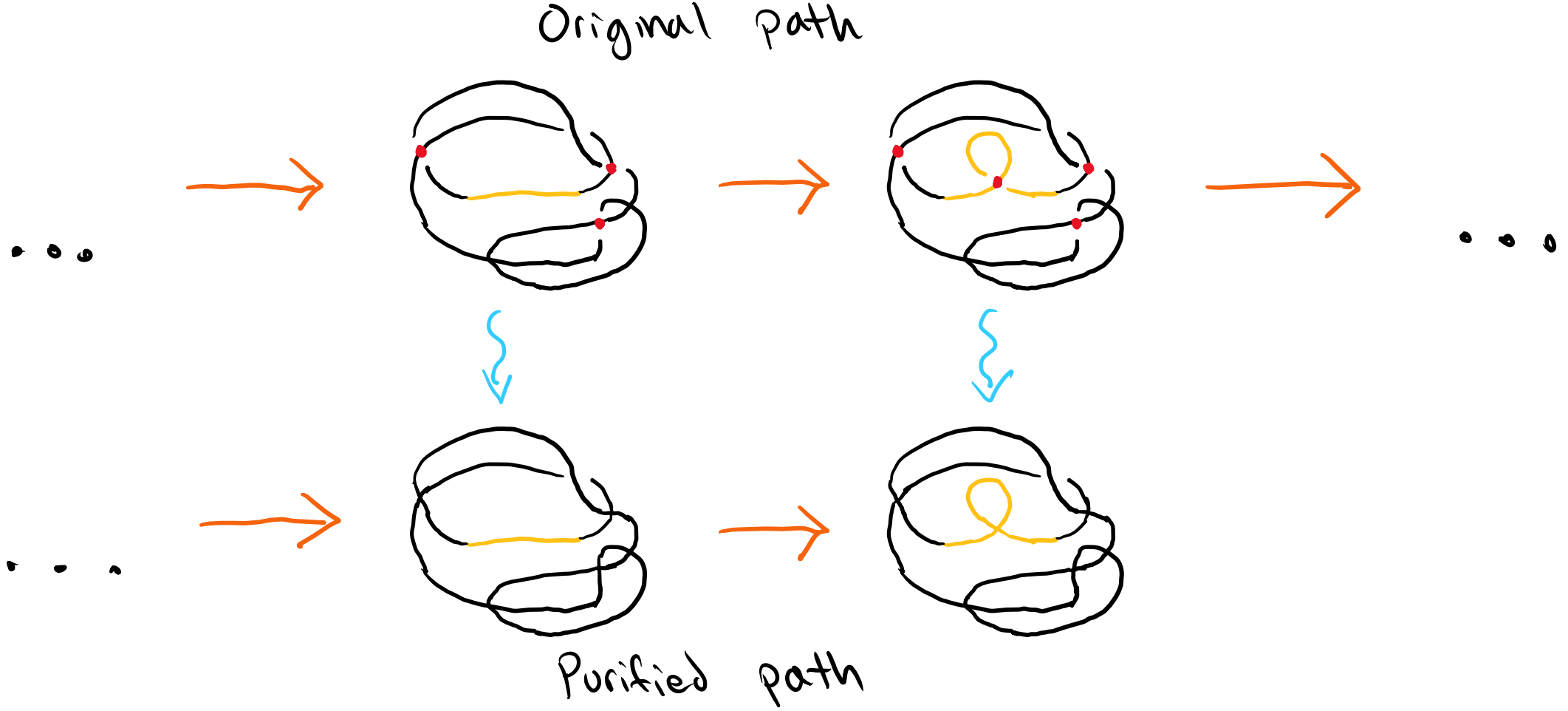
Judgment Rules

These rules are natural to impose, and generally result in our purified path extending.



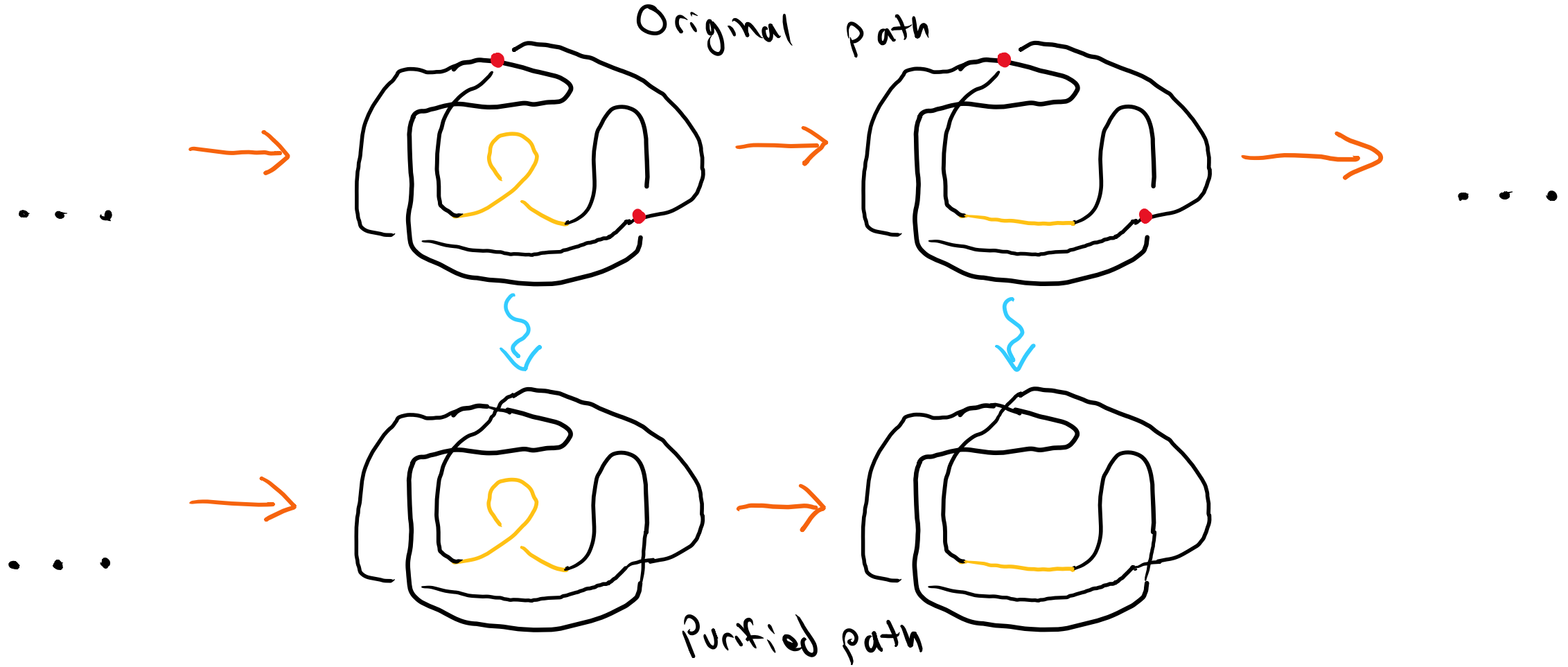
Judgment Rules

These rules are natural to impose, and generally result in our purified path extending.



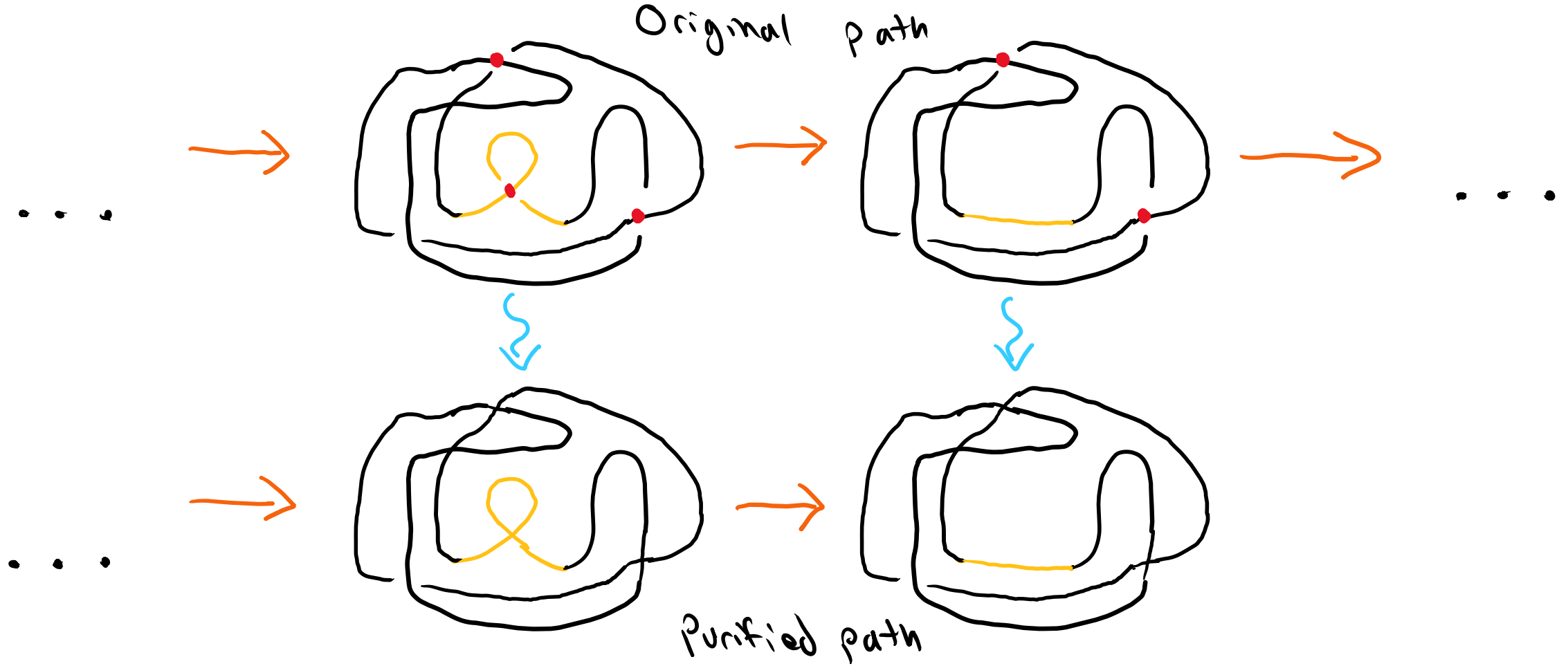
Judgment Rules

These rules are natural to impose, and generally result in our purified path extending.



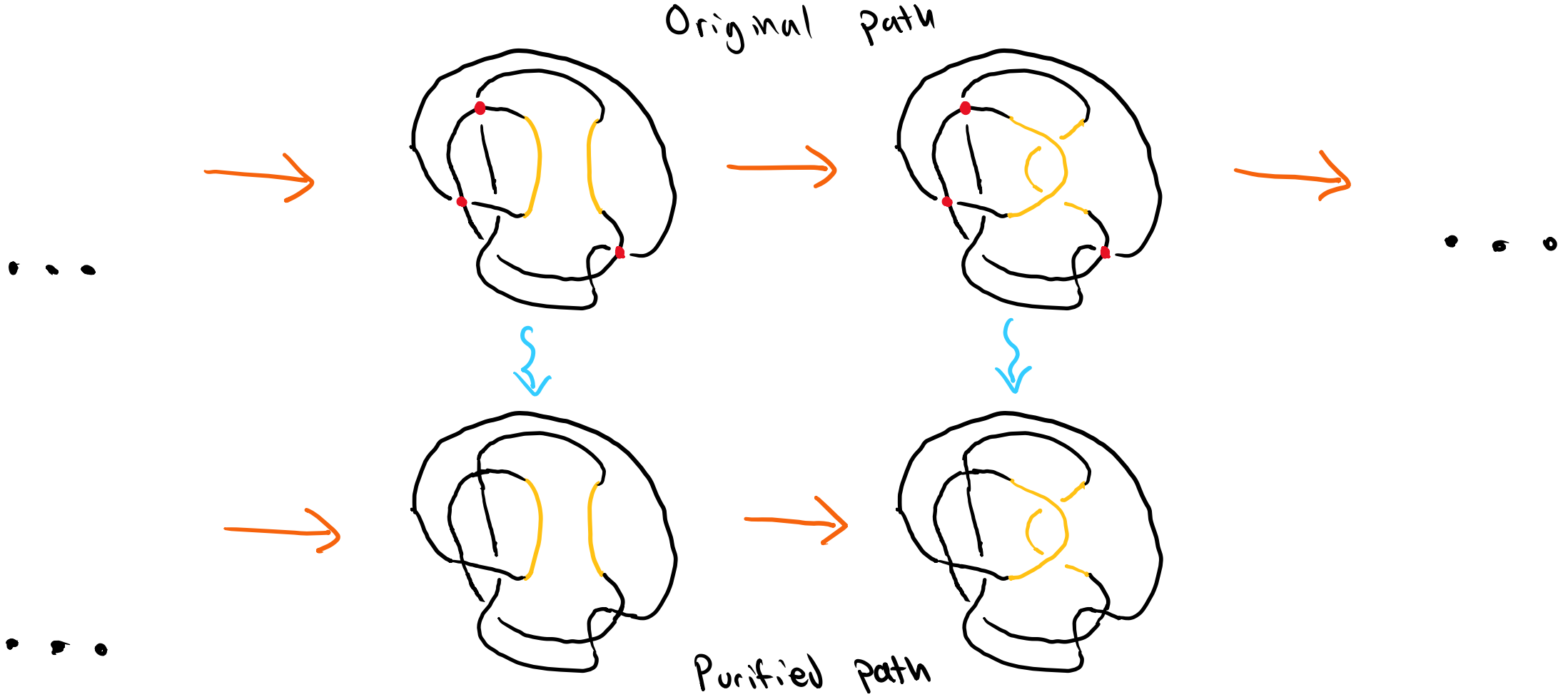
Judgment Rules

These rules are natural to impose, and generally result in our purified path extending.



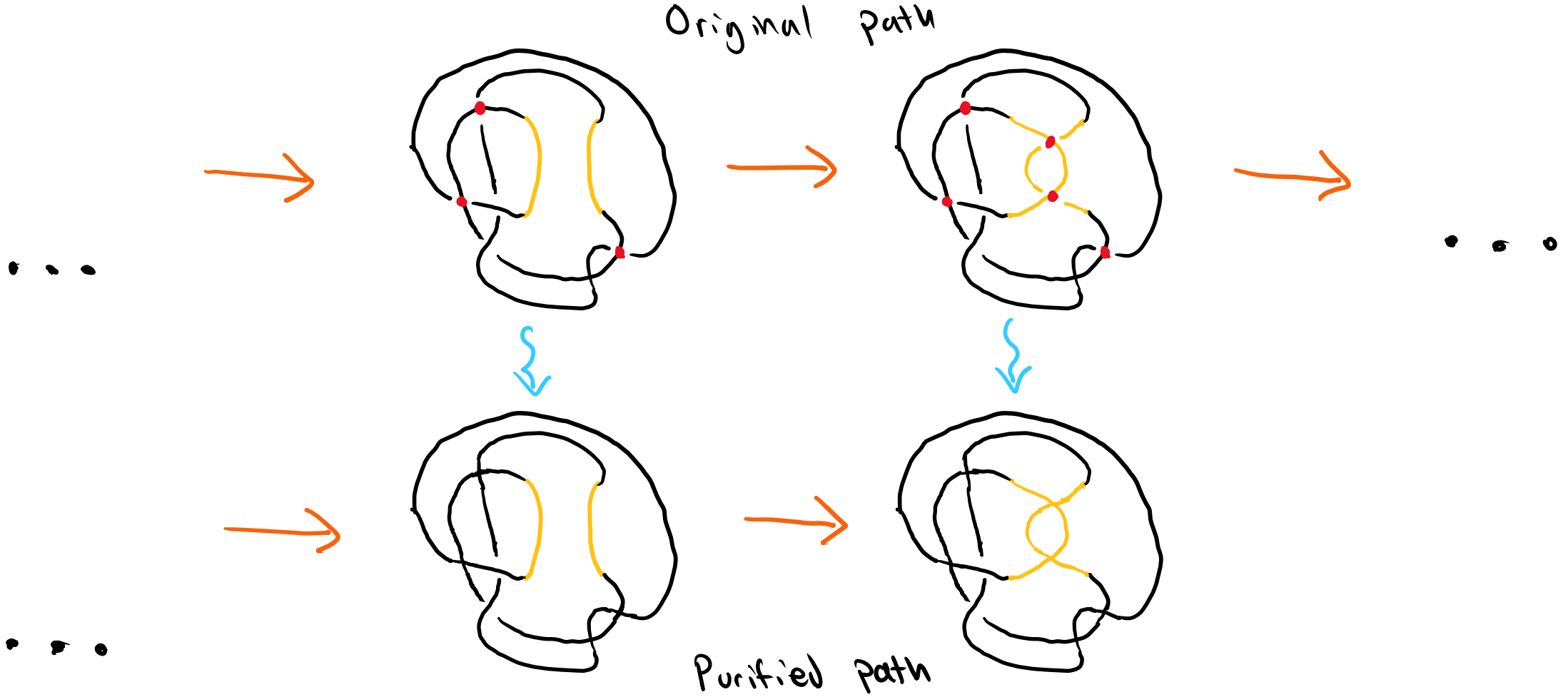
Judgment Rules

These rules are natural to impose, and generally result in our purified path extending.



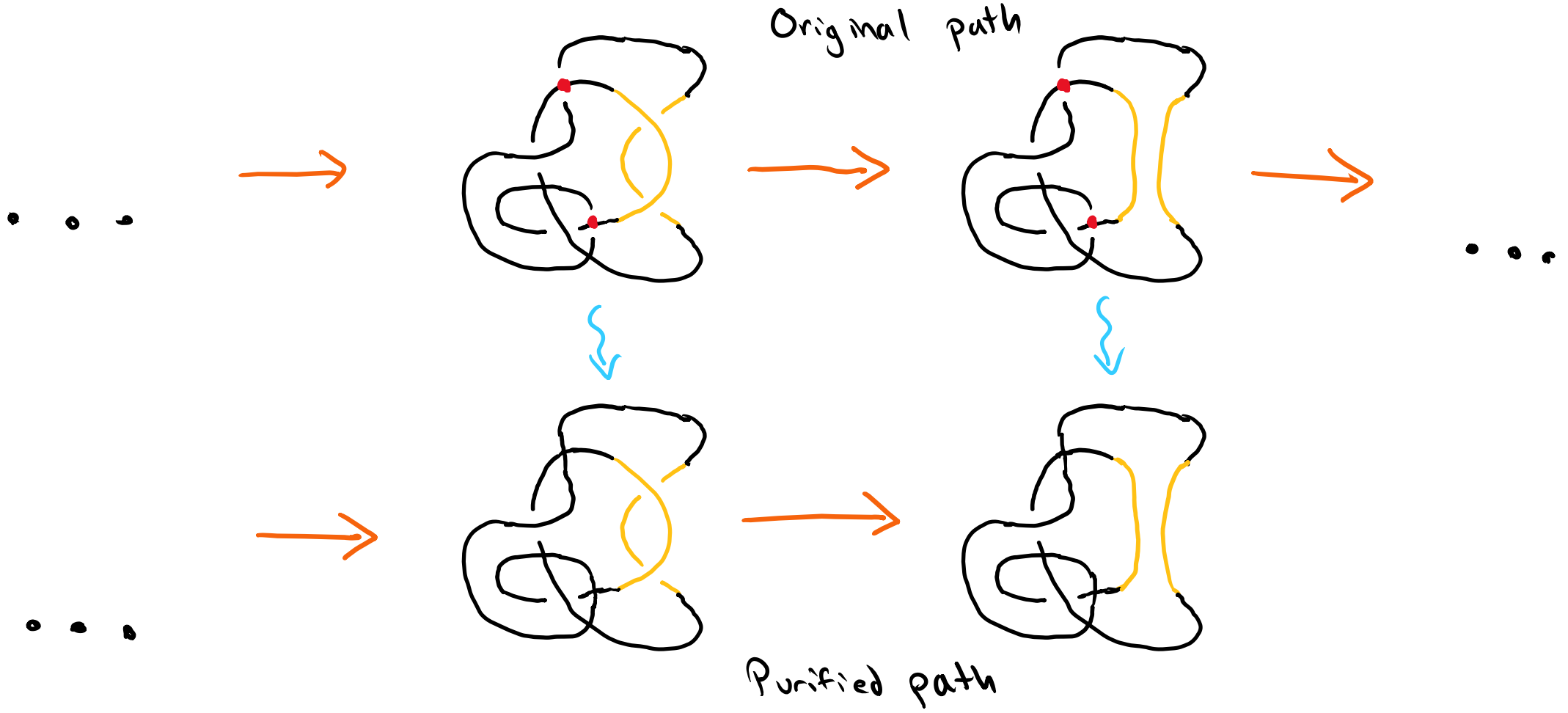
Judgment Rules

These rules are natural to impose, and generally result in our purified path extending.



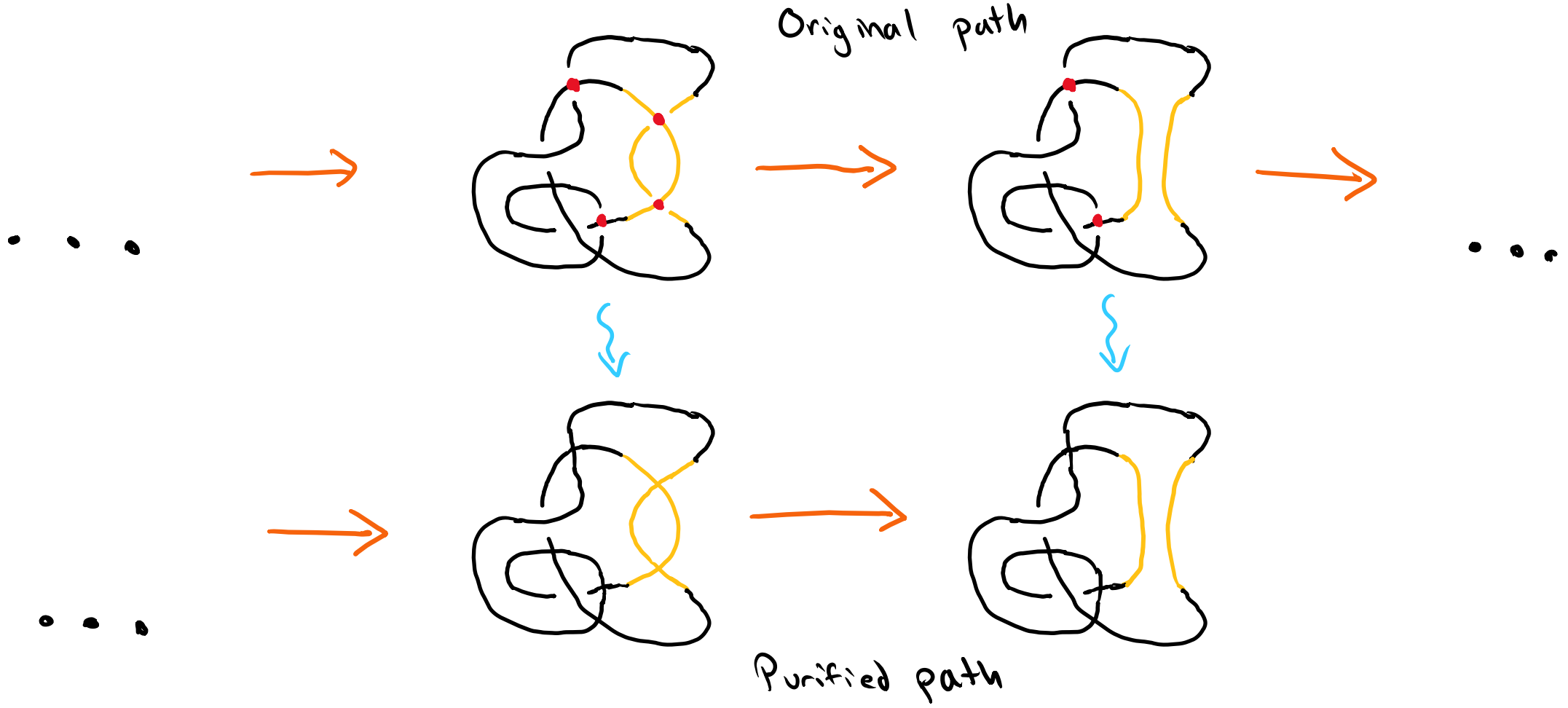
Judgment Rules

These rules are natural to impose, and generally result in our purified path extending.



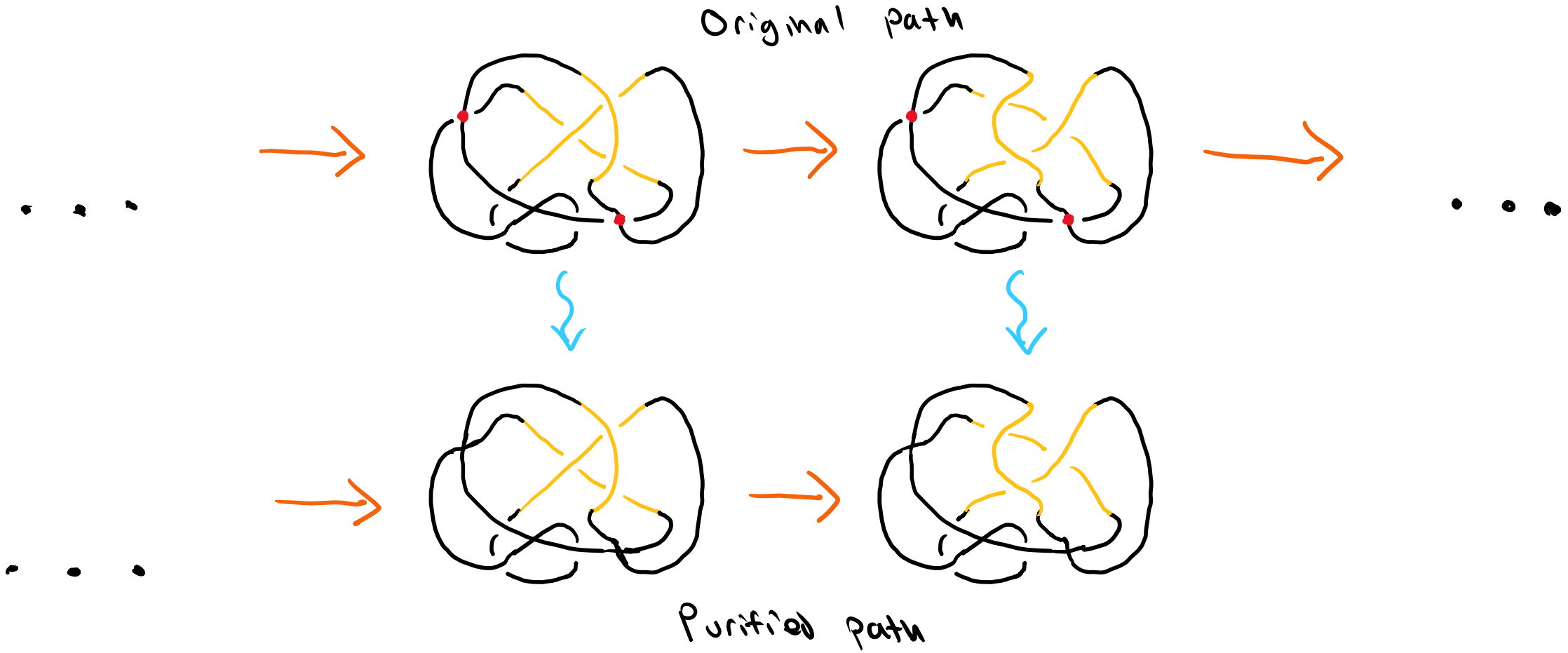
Judgment Rules

These rules are natural to impose, and generally result in our purified path extending.



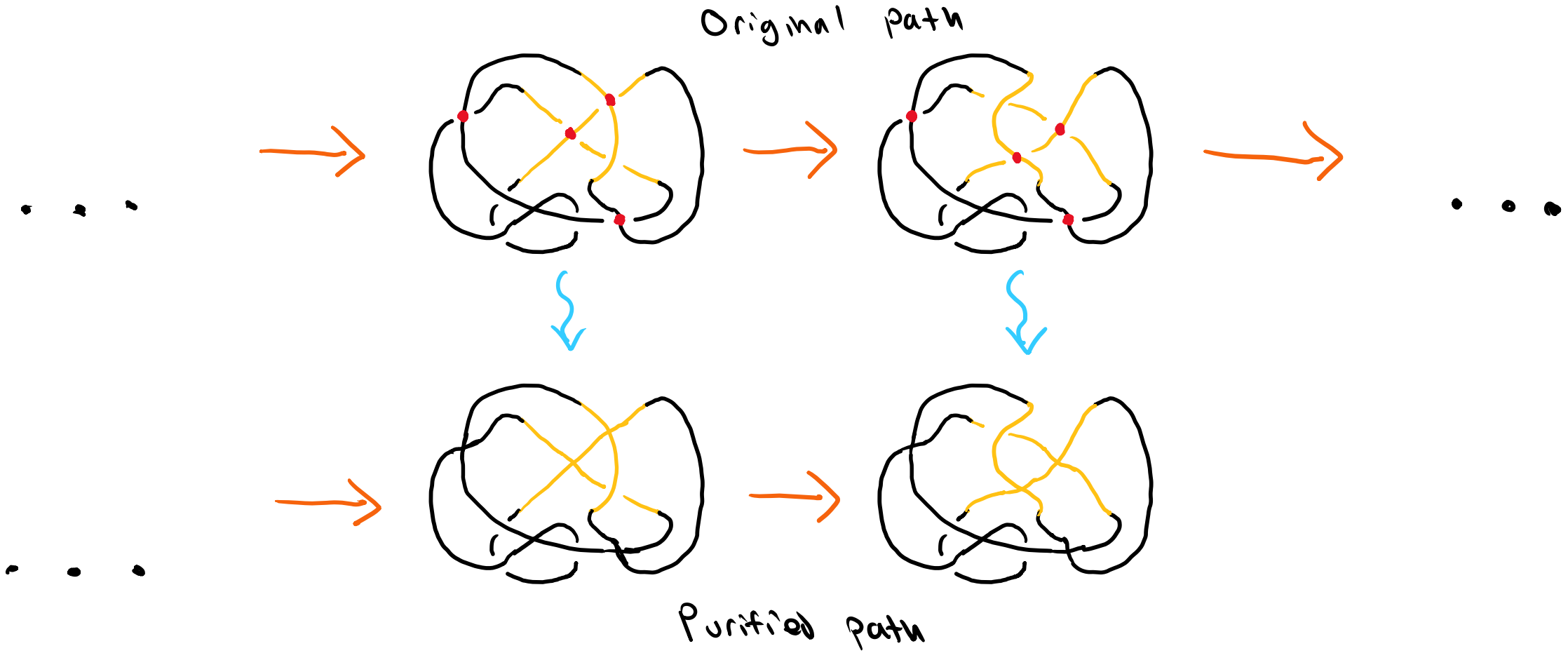
Judgment Rules

These rules are natural to impose, and generally result in our purified path extending.



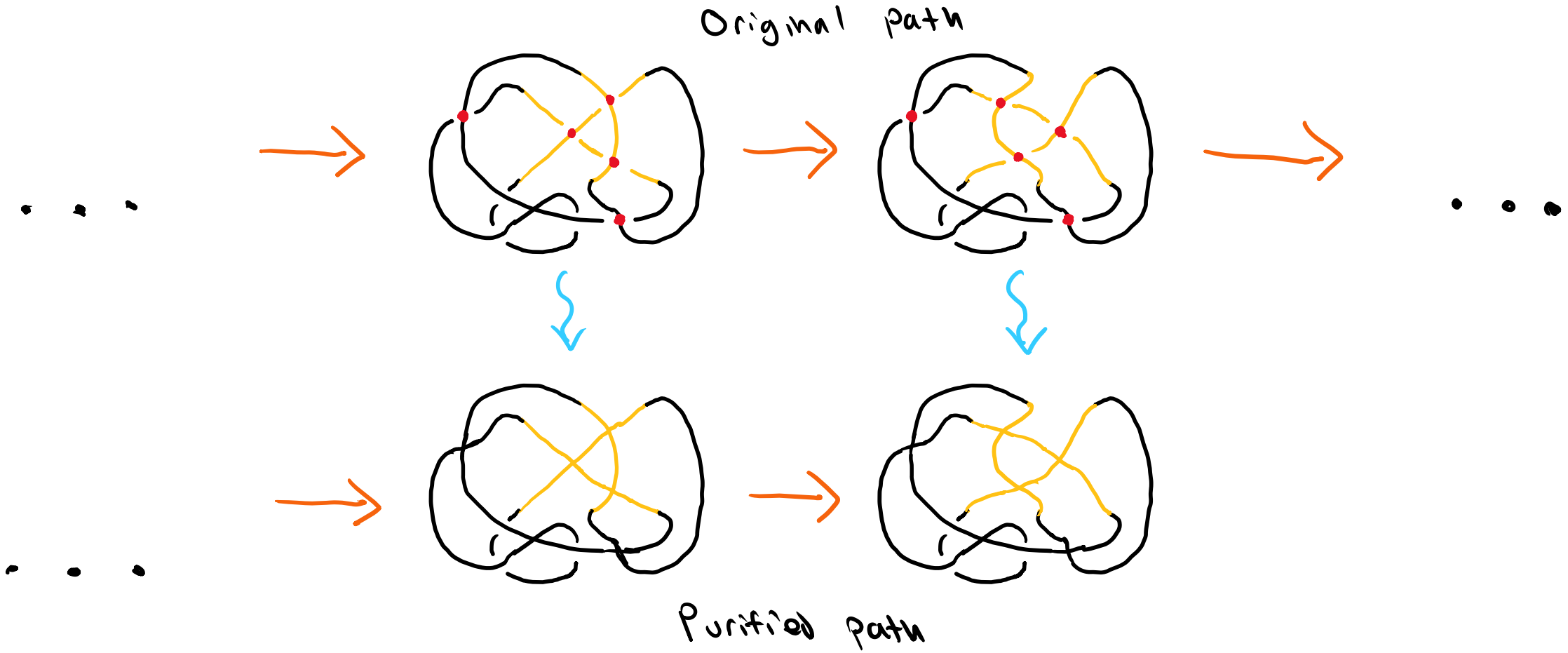
Judgment Rules

These rules are natural to impose, and generally result in our purified path extending.



Judgment Rules

These rules are natural to impose, and generally result in our purified path extending.



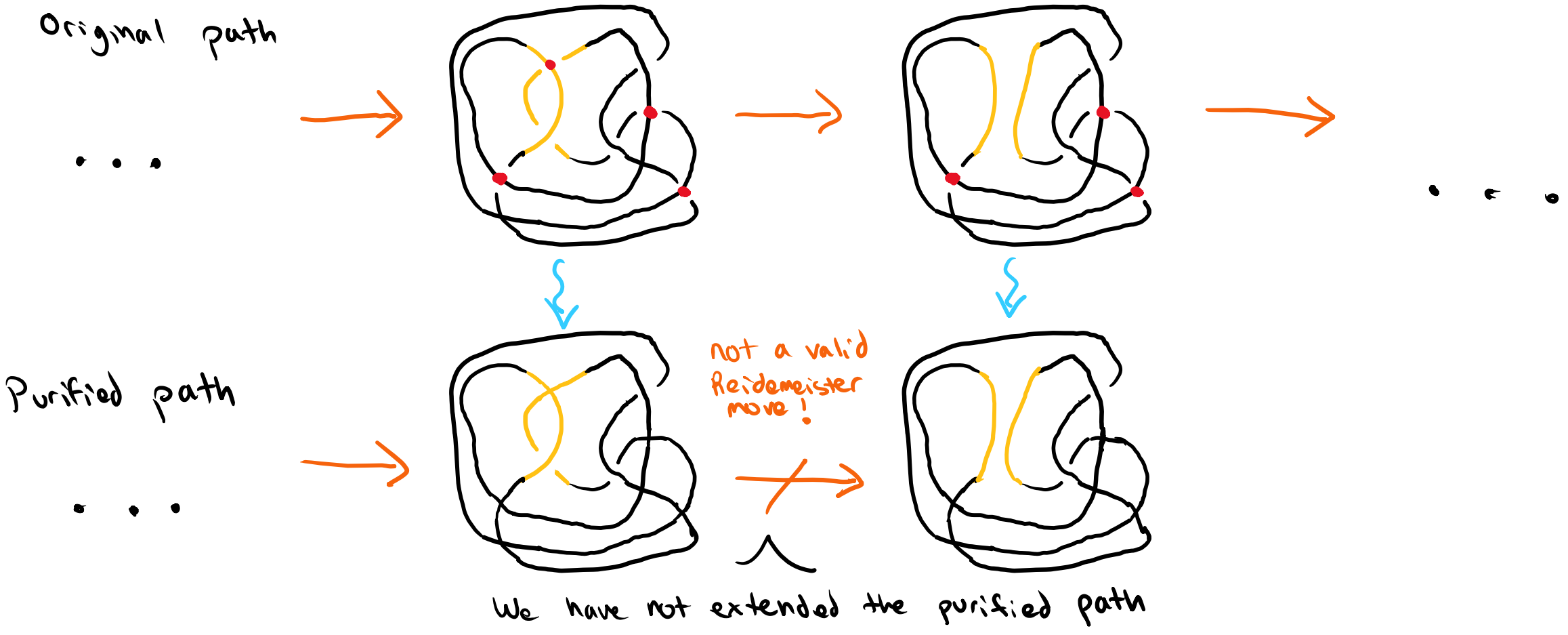
A Problem: Dead Ends

However, it is possible to run into a **dead end**, i.e., a situation where there is no clear way to choose the bad crossings for the next diagram in our plateau so that our purified path extends.

The two possible kinds of dead ends.

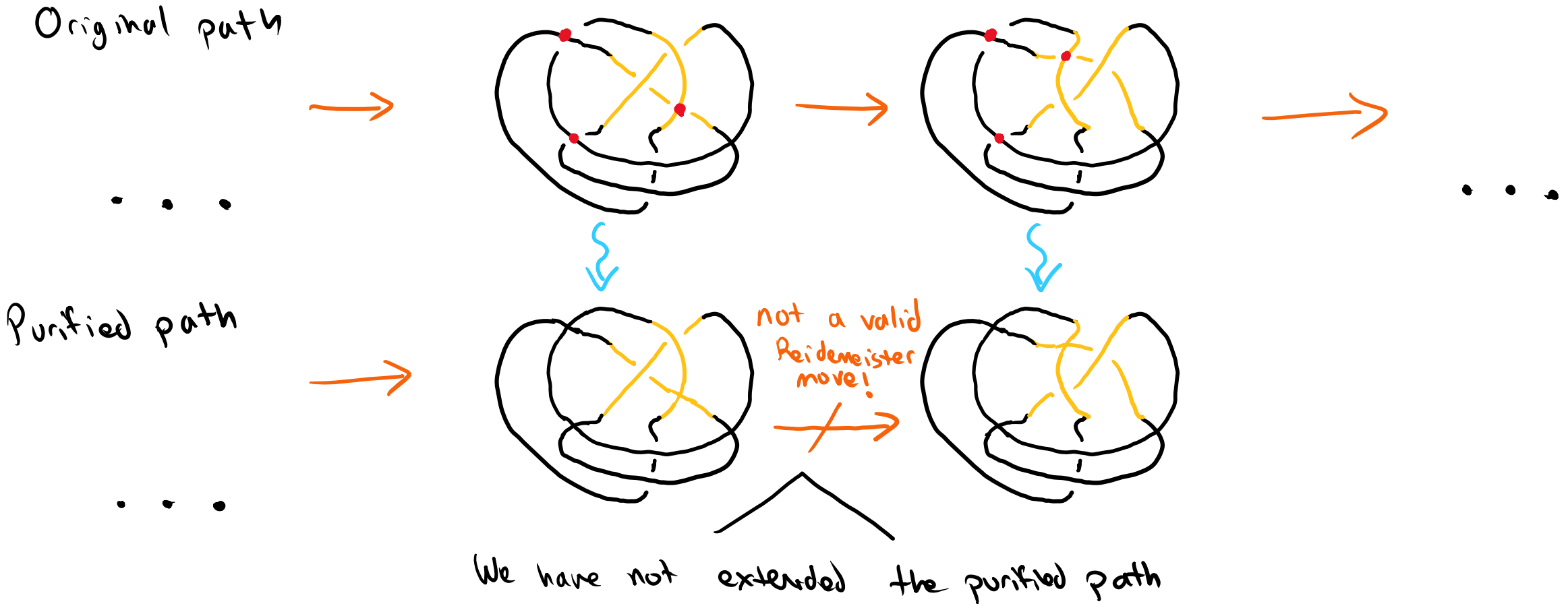
A Problem: Dead Ends

However, it is possible to run into a **dead end**, i.e., a situation where there is no clear way to choose the bad crossings for the next diagram in our plateau so that our purified path extends.



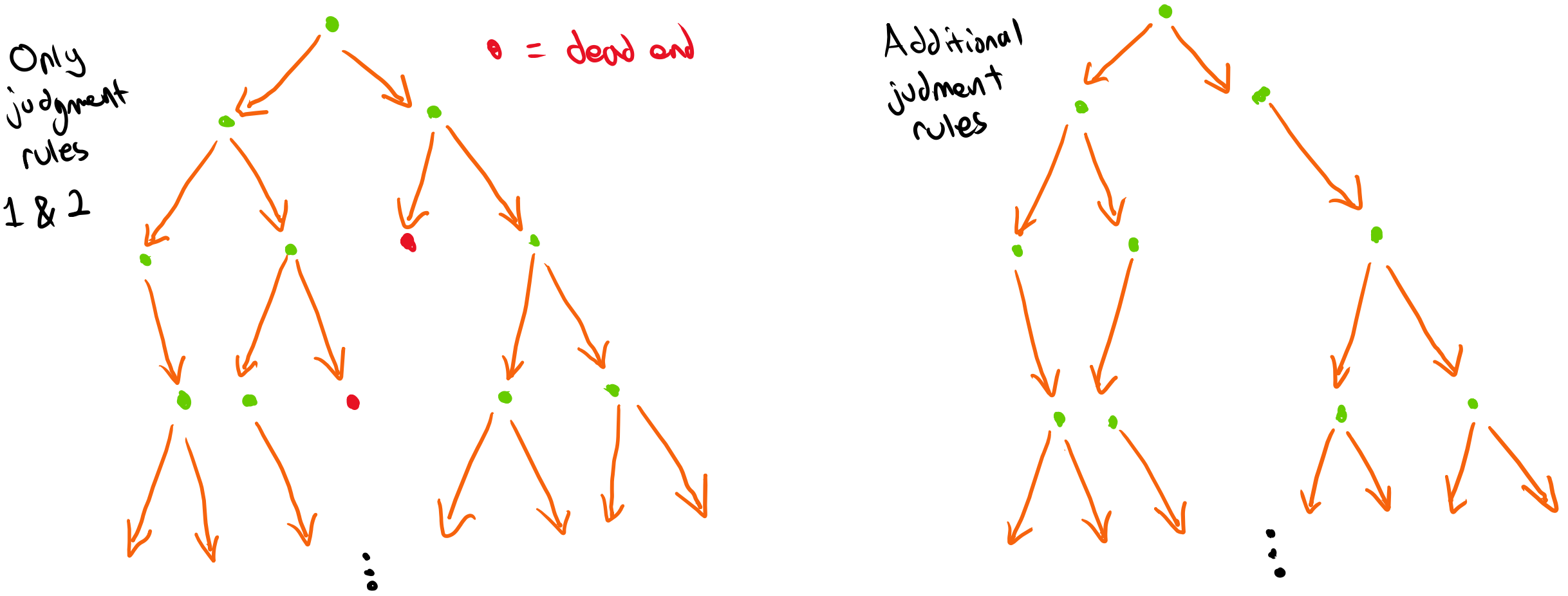
A Problem: Dead Ends

However, it is possible to run into a **dead end**, i.e., a situation where there is no clear way to choose the bad crossings for the next diagram in our plateau so that our purified path extends.



The Solution: Trim Our Decision Tree

We need additional judgment rules to trim the parts of our decision tree containing dead ends.

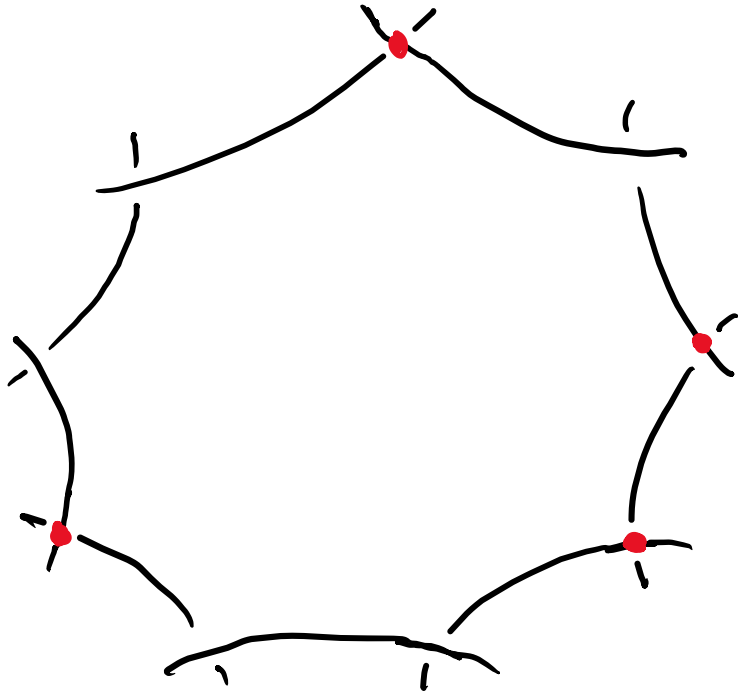


The Solution: Trim Our Tree

We add an additional judgment rule.

3. Any bounding cycle should contain an even number of bad crossings.

That is, our set of bad crossings should always be **even-bounding.**



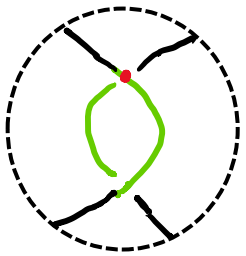
This bounding cycle has 4 bad crossings.

We require every bounding cycle to have an even number of bad crossings.

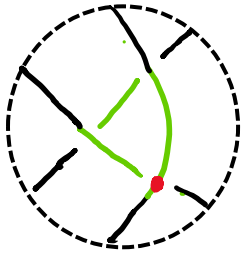
Even-Bounding Sets

If we follow judgment rule 3, we will never run into dead ends, as explained below.

It only remains to show that we can actually follow all our judgment rules.



Impossible since the green edges form a bounding cycle

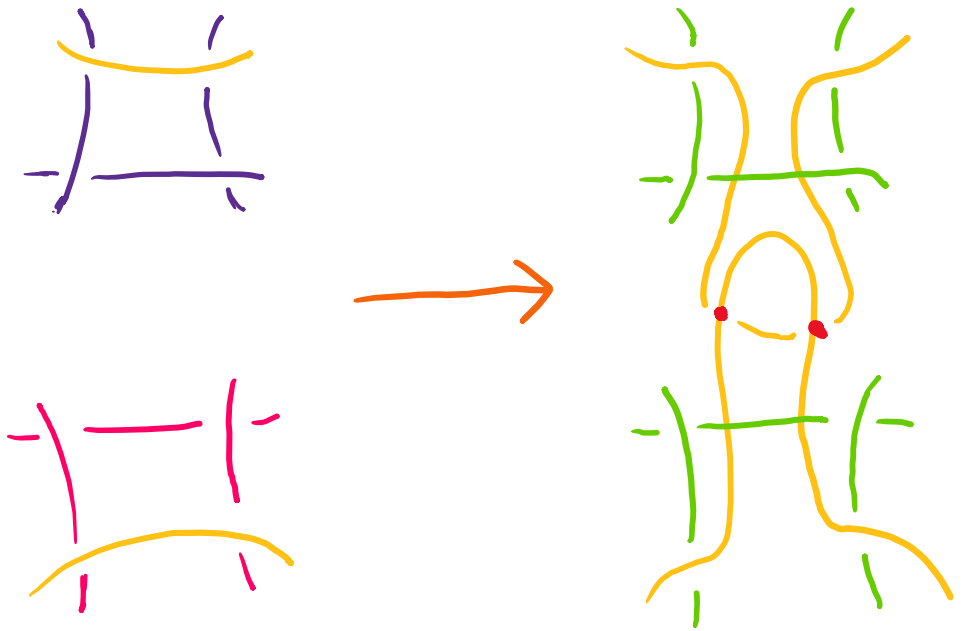


Impossible since the green edges form a bounding cycle

Even-Bounding Sets

For the first diagram A_j, we again choose the set of bad crossings to be empty, which is trivially even-bounding.

For the second diagram A_{j+1}, we again choose the two crossings introduced by the non-local forwards VR-II move M_{j+1} to form our bad set. This set is even-bounding, as shown below.

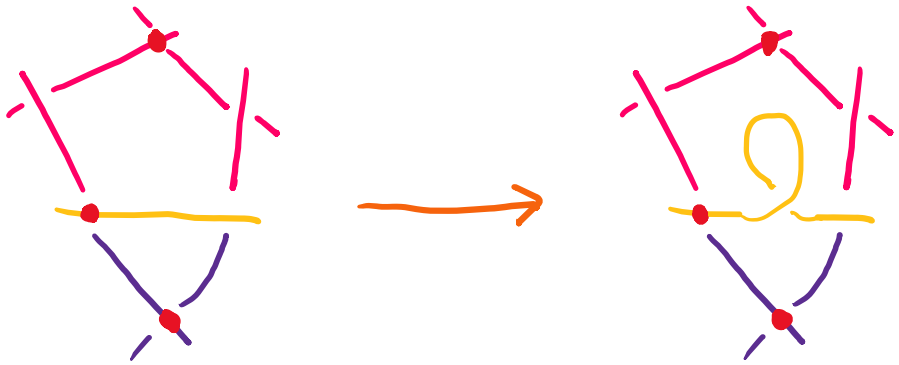


Every bounding cycle contains either 0 or 2 bad crossings

Even-Bounding Sets: Forwards VR-I

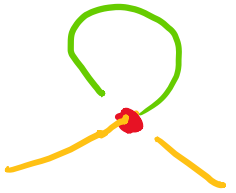
When the current move is a forwards VR-I move, we follow judgment rule 1 and declare the crossing introduced by the move to be good in compliance with judgment rule 2.

The result is an even-bounding set for the next diagram, in compliance with judgment rule 3.



- pink cycle same number of bad crossings
- purple cycle same number of bad crossings

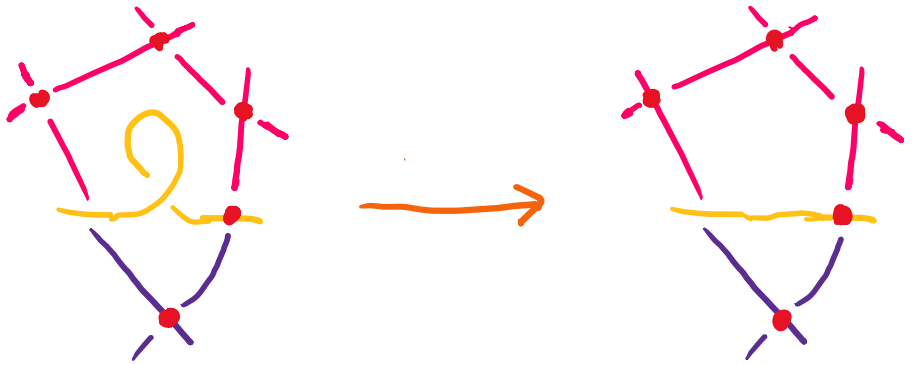
★ Cannot classify the crossing created by the move as good since the loop is a bounding cycle.



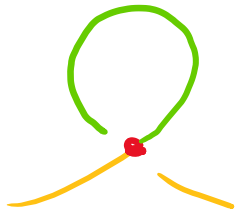
Even-Bounding Sets: Backwards VR-I

When the current move is a backwards VR-I move, following judgment rule 1 uniquely determines the set of bad crossings.

The result is an even-bounding set for the next diagram, in compliance with judgment rule 3.



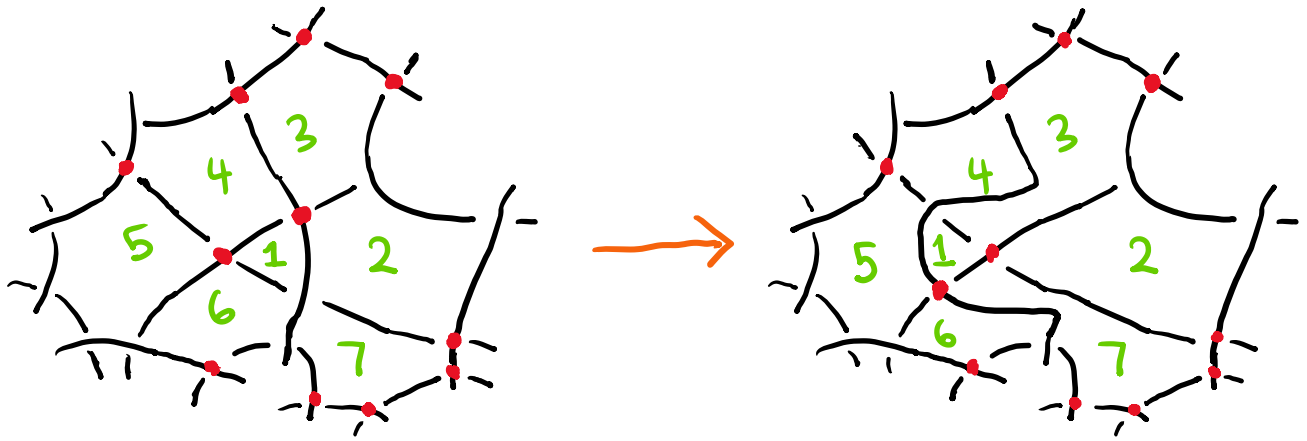
- pink cycle same number of bad crossings
- purple cycle same number of bad crossings

★ Note  is impossible since the loop is a bounding cycle

Even-Bounding Sets: VR-III

When the current move is a VR-III move, following judgment rule 1 uniquely determines the set of bad crossings.

The result is an even-bounding set for the next diagram, in compliance with judgment rule 3.



The number of bad crossings in each of the bounding cycles 2 to 7 changes by the number of bad crossings in the bounding cycle 1 mod two.

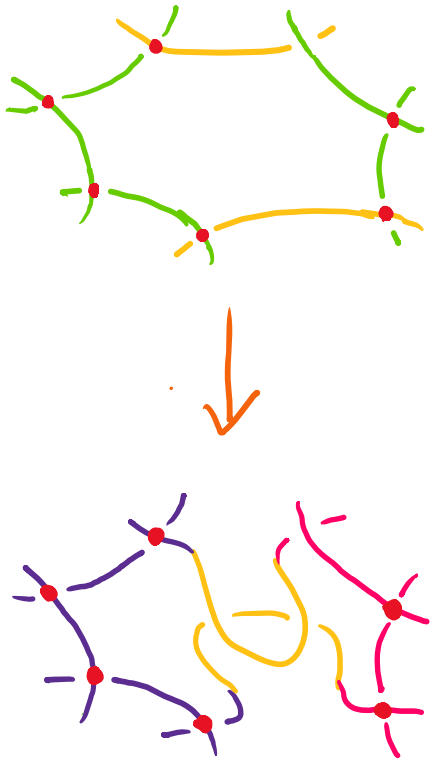
Even-Bounding Sets: Local Forwards VR-II

When the current move is a local forwards VR-II move, we follow judgment rule 1, and of the two choices in compliance with judgment rule 2, one option is always even-bounding, as shown below.

Therefore, we have an even-bounding set for the next diagram, in compliance with judgment rule 3.

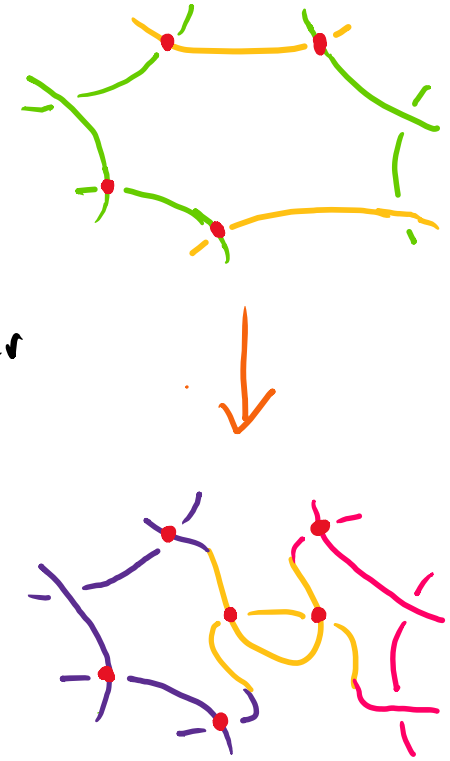
Case 1:

If the pink and purple sections each have an even number of bad crossings, the two crossings created by the move are classified as good.



Case 2:

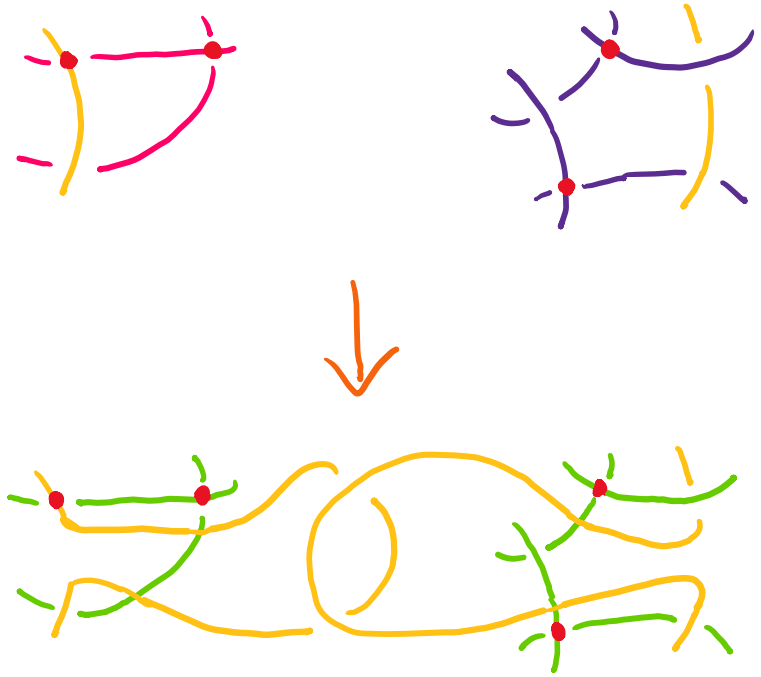
If the pink and purple sections each have an odd number of bad crossings, the two crossings created by the move are classified as bad.



Even-Bounding Sets: Non-Local Forwards VR-II

When the current move is a non-local forwards VR-II move, we follow judgment rule 1, and therefore have two choices in compliance with judgment rule 2: declare both crossings introduced by the move to be bad, or declare both to be good.

Both choices result in an even-bounding set for the next diagram, in compliance with judgment rule 3.



Number of bad crossings in green cycle

= Number of bad crossings in pink cycle

+ Number of bad crossings in purple cycle

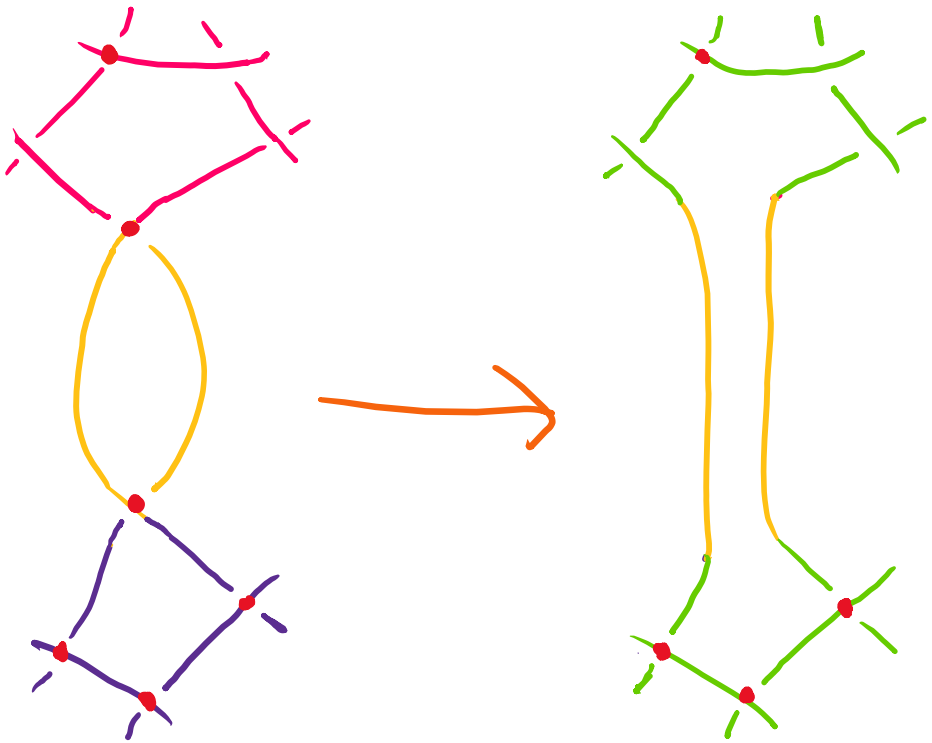
+ Number of bad crossings created by the move

even since we follow rule 2

Even-Bounding Sets: Local Backwards VR-II

When the current move is a local backwards VR-II move, following judgment rule 1 uniquely determines the set of bad crossings.

The result is an even-bounding set for the next diagram, in compliance with judgment rule 3.



number of bad crossings in green cycle

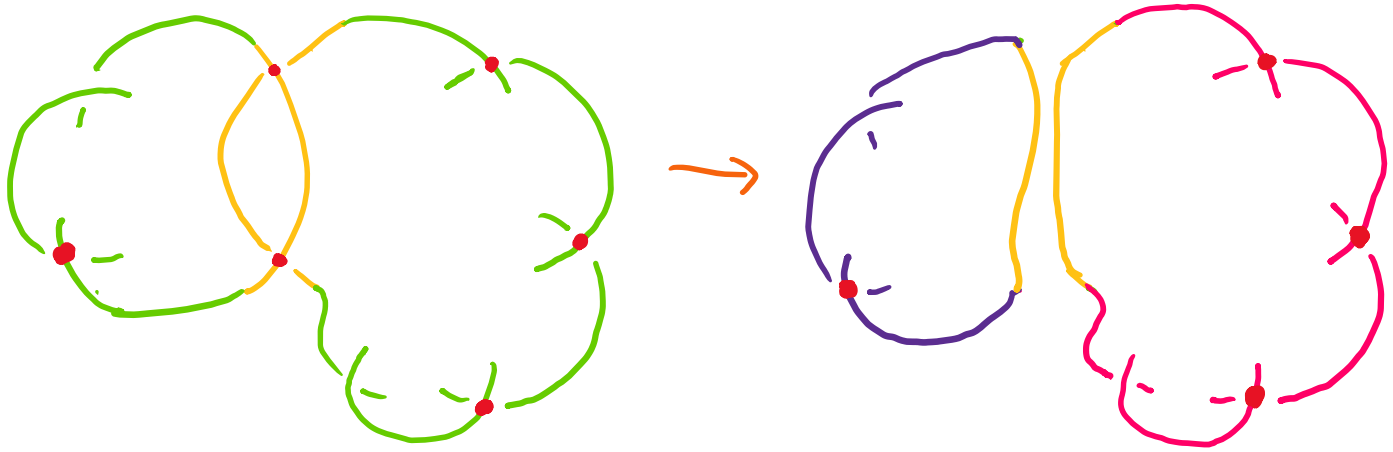
= number of bad crossings in pink cycle

+ number of bad crossings in purple cycle

− number of bad crossings in yellow cycle

Even-Bounding Sets: Non-Local Backwards VR-II

When the current move is a local backwards VR-II move, following judgment rule 1 uniquely determines the set of bad crossings. There is however no reason for the resulting set to be even-bounding.



All we can say is

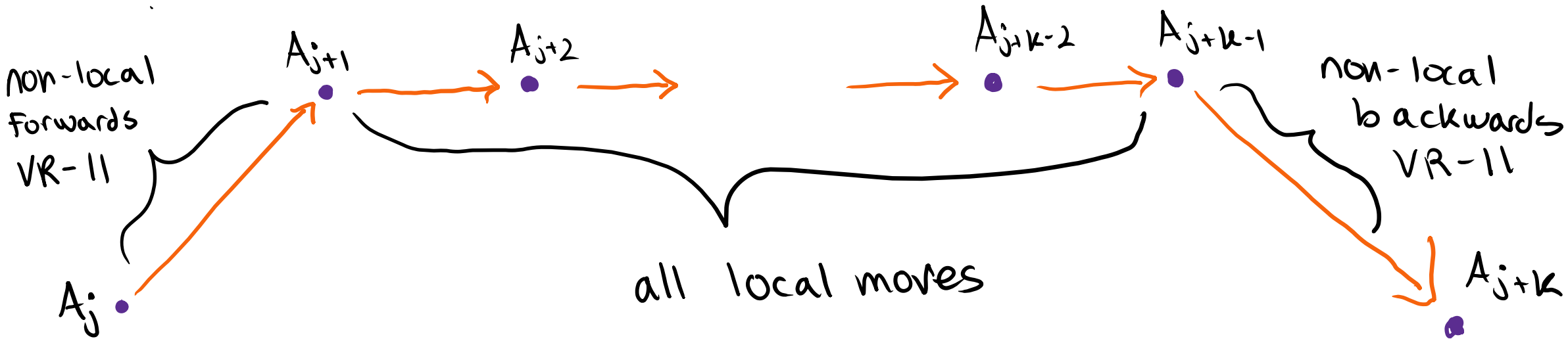
number of bad crossings in green cycle
− number of bad crossings in yellow cycle
= number of bad crossings in pink cycle
+ number of bad crossings in purple cycle

Purifying our Plateau

Though there is an issue when applying a non-local backwards VR-II move, by definition the only such move occurs at the end of our plateau, so all our bad sets are guaranteed to be even-bounding except the last, which means we still never run into a dead end.

Therefore, we can purify our plateau to obtain a less complicated path $(A_j - X_0, A_{j+1} - X_1, \dots, A_{j+k} - X_k)$, where X_0 is empty.

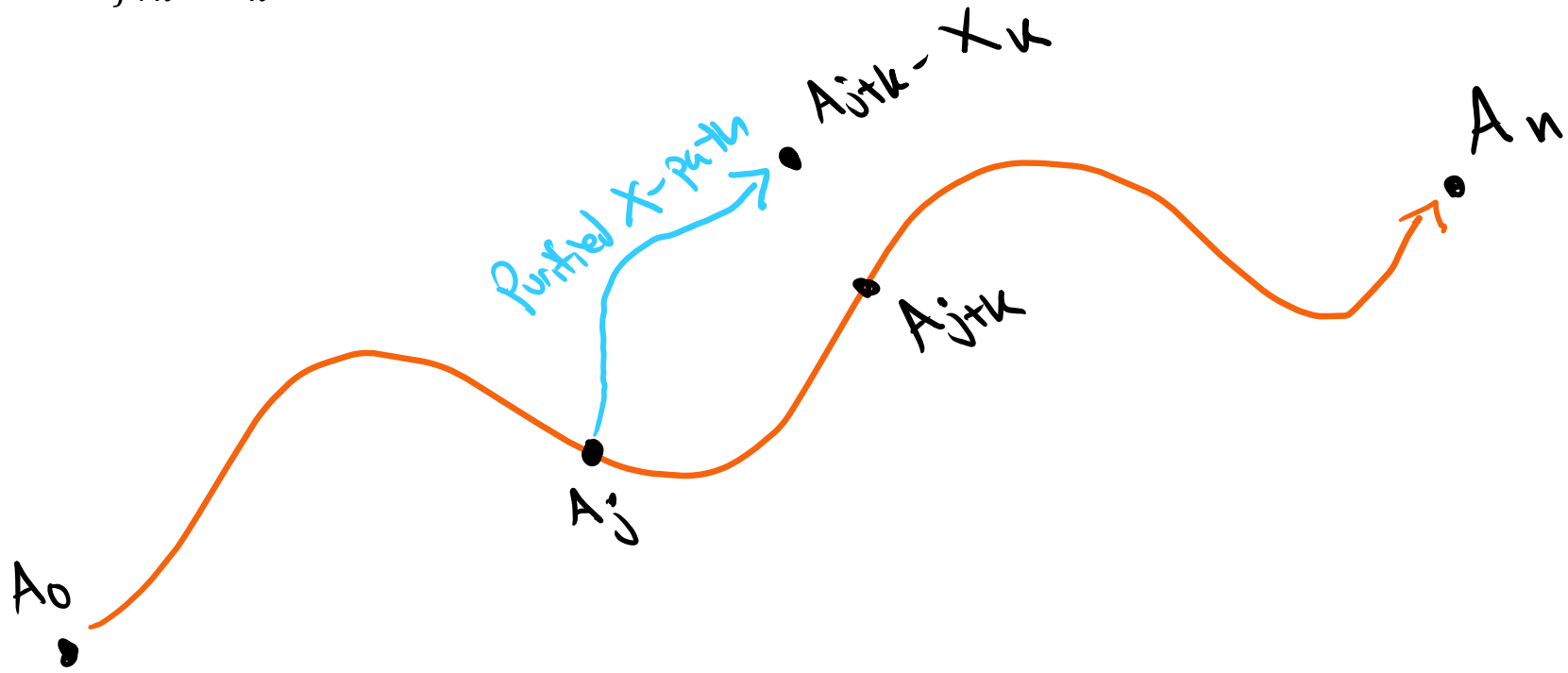
Recall that $A_{j+i} - X_i$ denotes the result of removing the crossings in X_i from A_{j+i}.



Another Problem: Preserving Endpoints

There is another issue. We wanted to replace our plateau $A_j \to_{M_{j+1}} \ldots \to_{M_{j+k}} A_{j+k}$, which is a subpath of $A_0 \to_{M_1} \ldots \to_{M_n} A_n$, with a less complicated path.

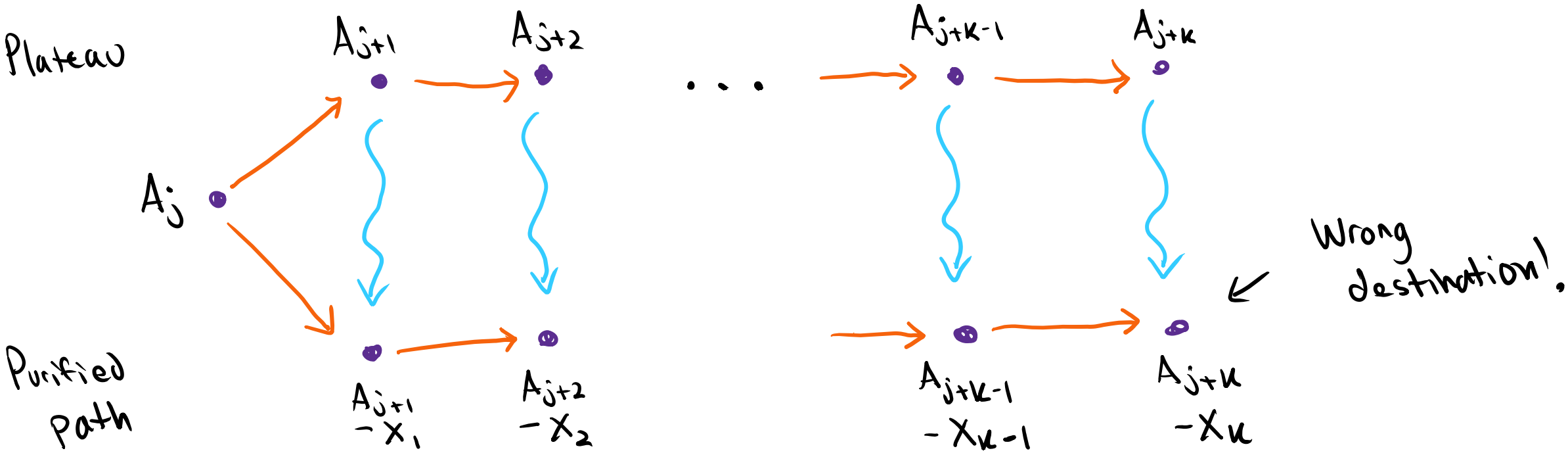
However, our alternative $(A_j, A_{j+1} - X_1, \ldots, A_{j+k} - X_k)$ cannot necessarily serve as a replacement, because it does not end at A_{j+k} if X_k is nonempty.



Another Problem: Preserving Endpoints

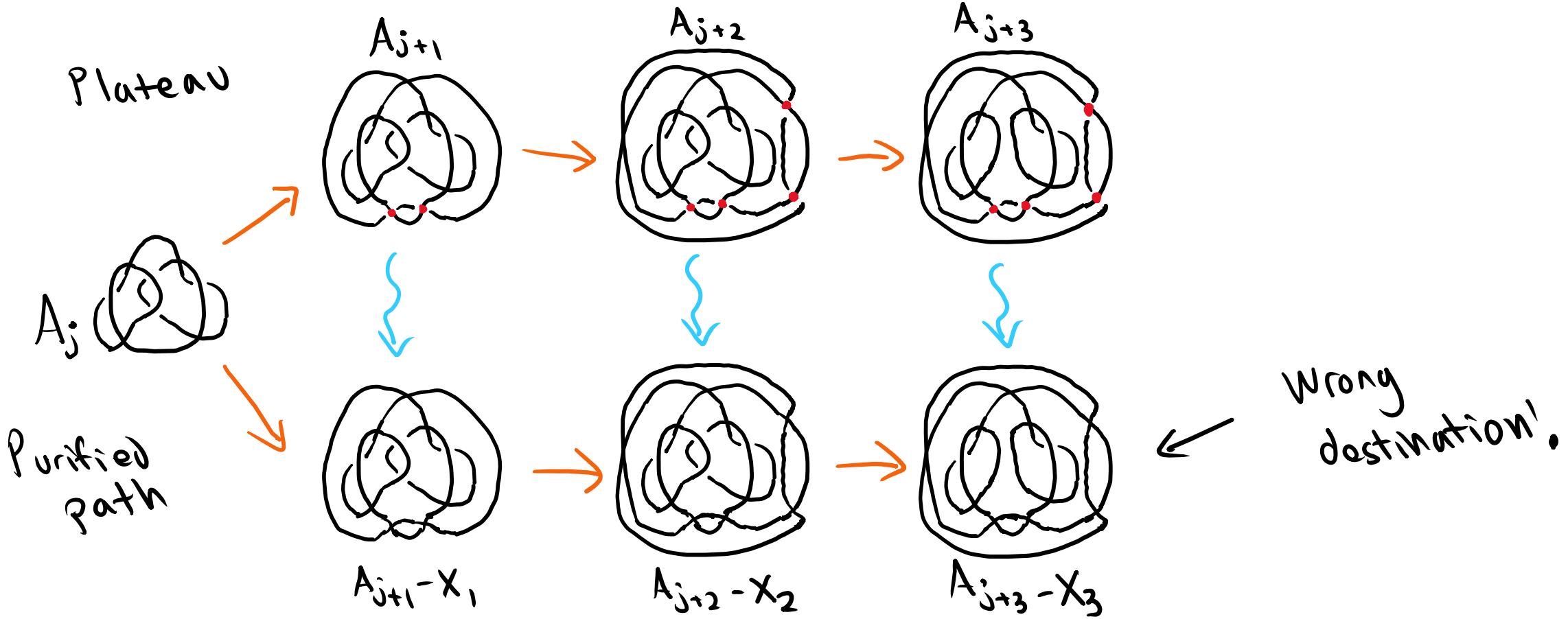
There is another issue. We wanted to replace our plateau $A_j \to_{M_{j+1}} \dots \to_{M_{j+k}} A_{j+k}$, which is a subpath of $A_0 \to_{M_1} \dots \to_{M_n} A_n$, with a less complicated path.

However, our alternative $(A_j, A_{j+1} - X_1, \dots, A_{j+k} - X_k)$ cannot necessarily serve as a replacement, because it does not end at A_{j+k} if X_k is nonempty.



Another Problem: Preserving Endpoints

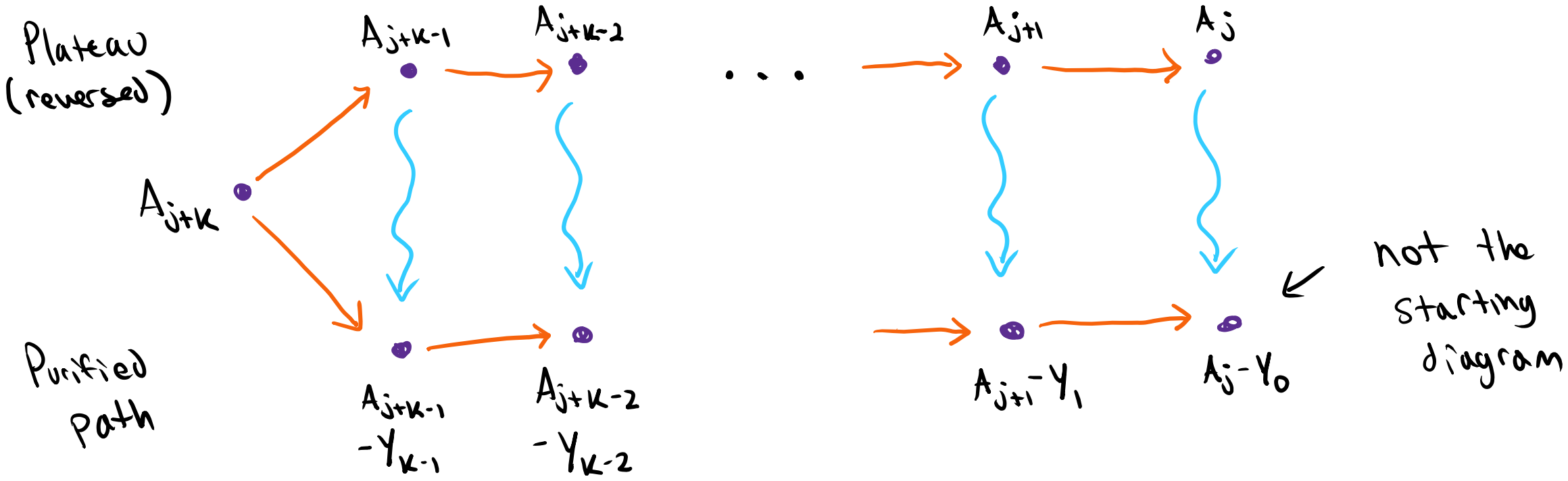
It is indeed possible for the last bad set X_k to be nonempty, as shown below.



Solution: Build More Paths

Specifically, the situation with our plateau is symmetric, so we can apply everything in reverse.

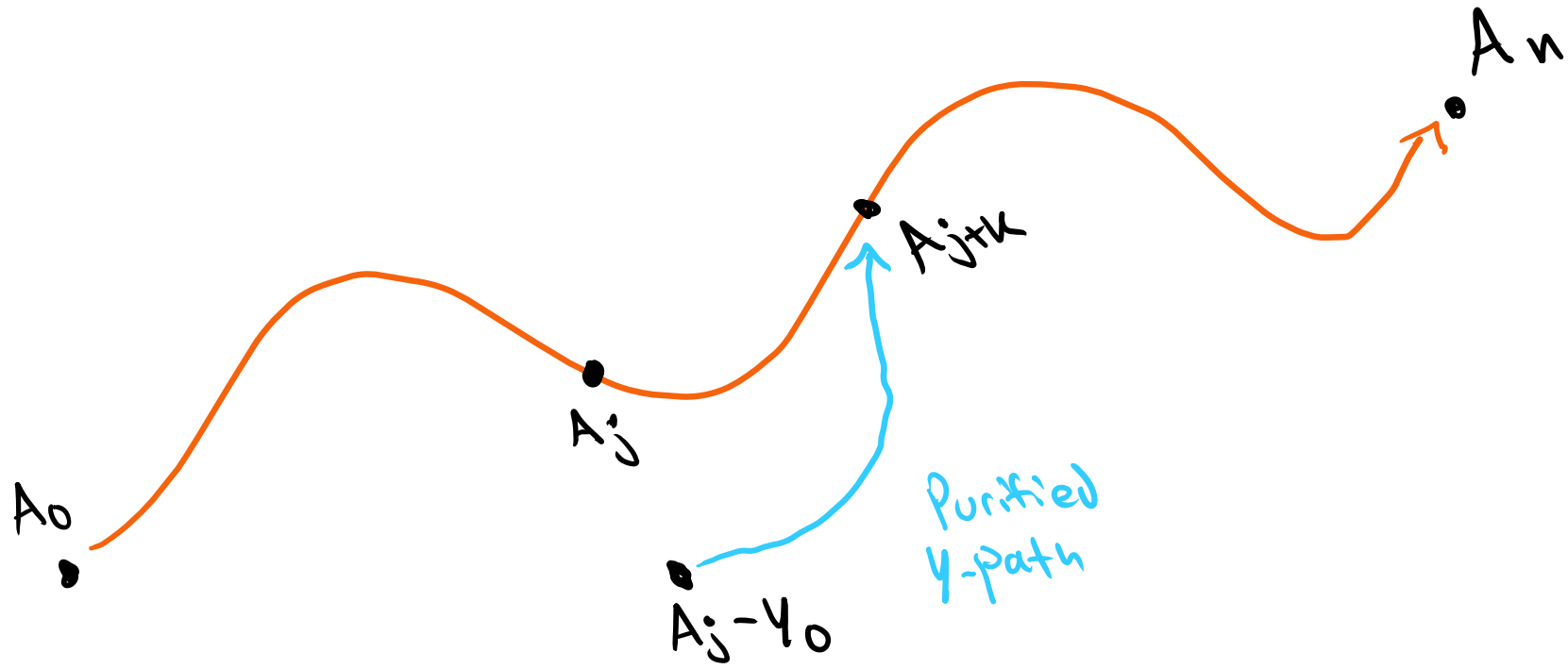
That is, we can find sets $Y_0, \ldots, Y_k$ such that Y_k is empty, Y_{k-1} is non-empty and $(A_j - Y_0, \ldots, A_{j+k} - Y_k)$ is a path.



Solution: Build More Paths

Specifically, the situation with our plateau is symmetric, so we can apply everything in reverse.

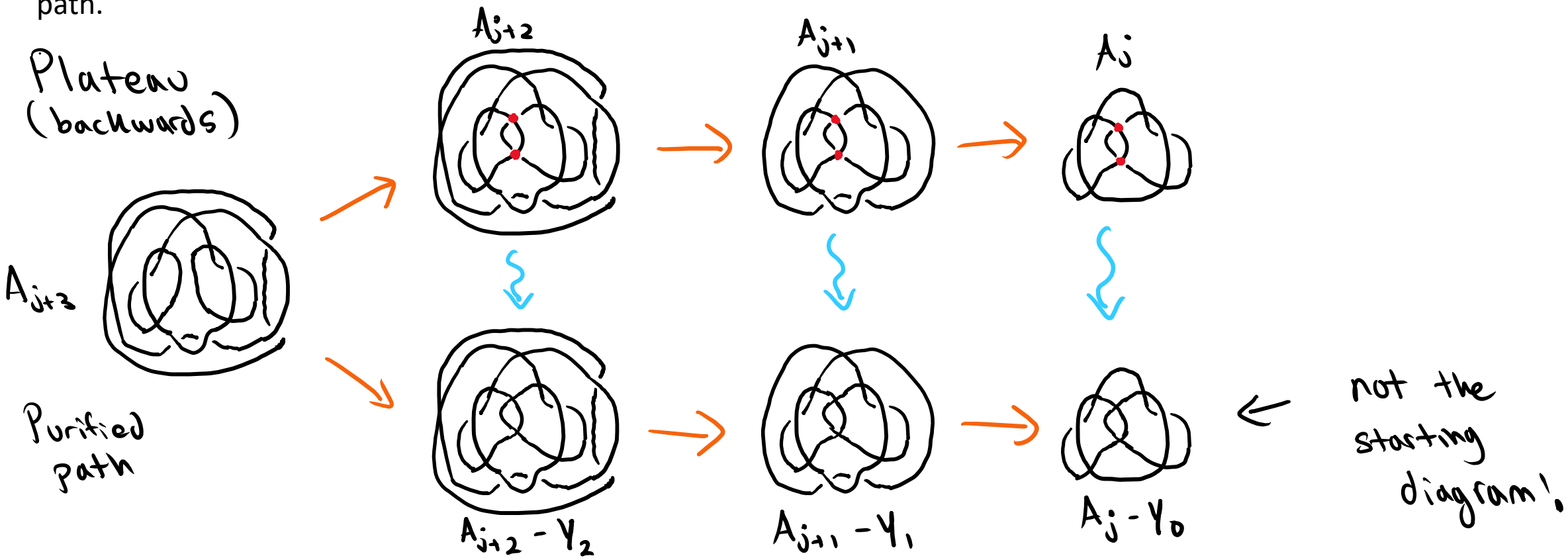
That is, we can find sets $Y_0, \ldots, Y_k$ such that Y_k is empty, Y_{k-1} is non-empty and $(A_j - Y_0, \ldots, A_{j+k} - Y_k)$ is a path.



Solution: Build More Paths

Specifically, the situation with our plateau is symmetric, so we can apply everything in reverse.

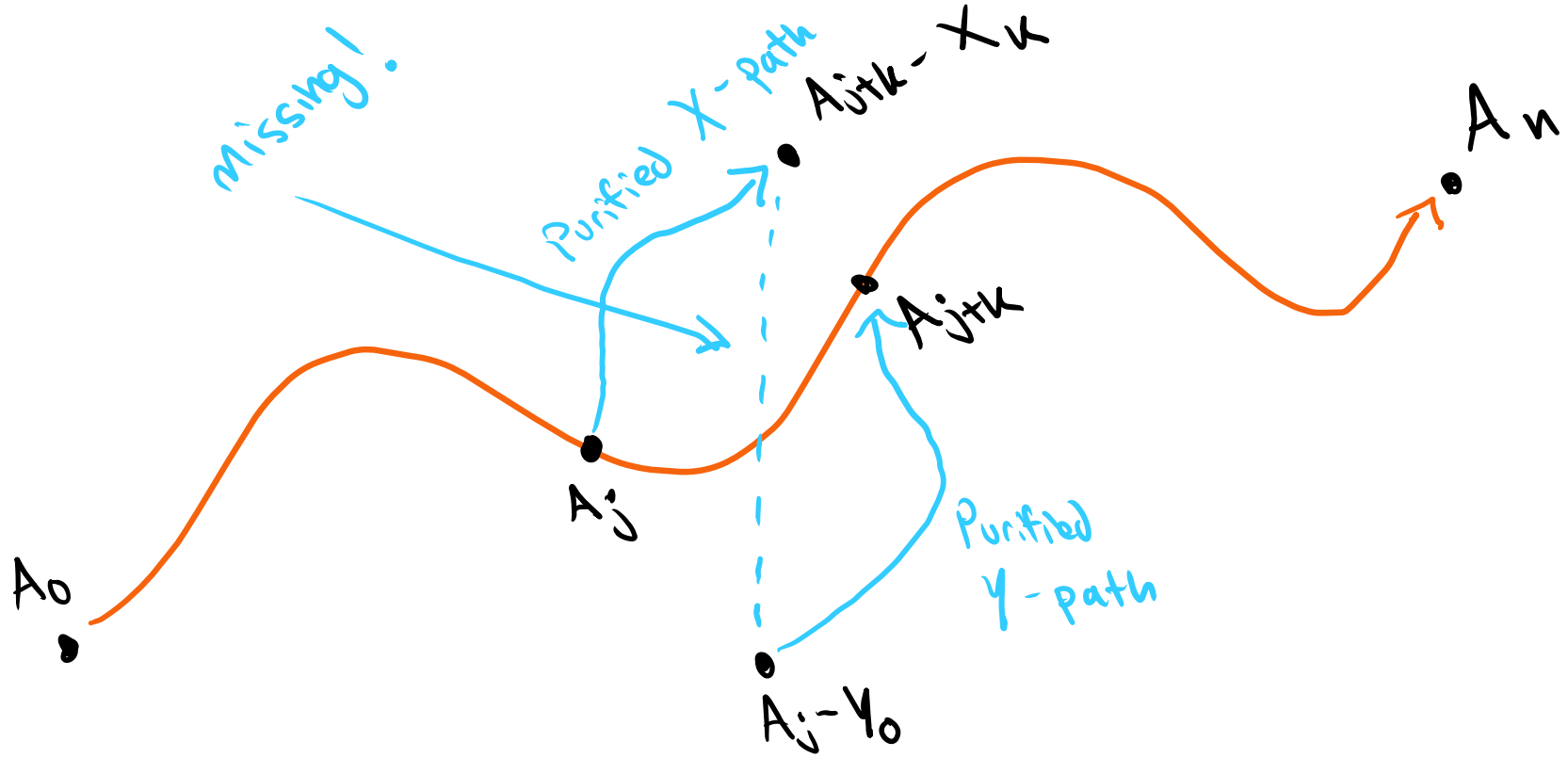
That is, we can find sets $Y_0, \ldots, Y_k$ such that Y_k is empty, Y_{k-1} is non-empty and $(A_j - Y_0, \ldots, A_{j+k} - Y_k)$ is a path.



Solution: Build More Paths

Now we have a path $(\mathrm{A_j}, \mathrm{A_{j+1}} - \mathrm{X_1}, \ldots, \mathrm{A_{j+k}} - \mathrm{X_k})$ and a path $(A_j - Y_0, \ldots, A_{j+k-1} - Y_{k-1}, A_{j+k})$.

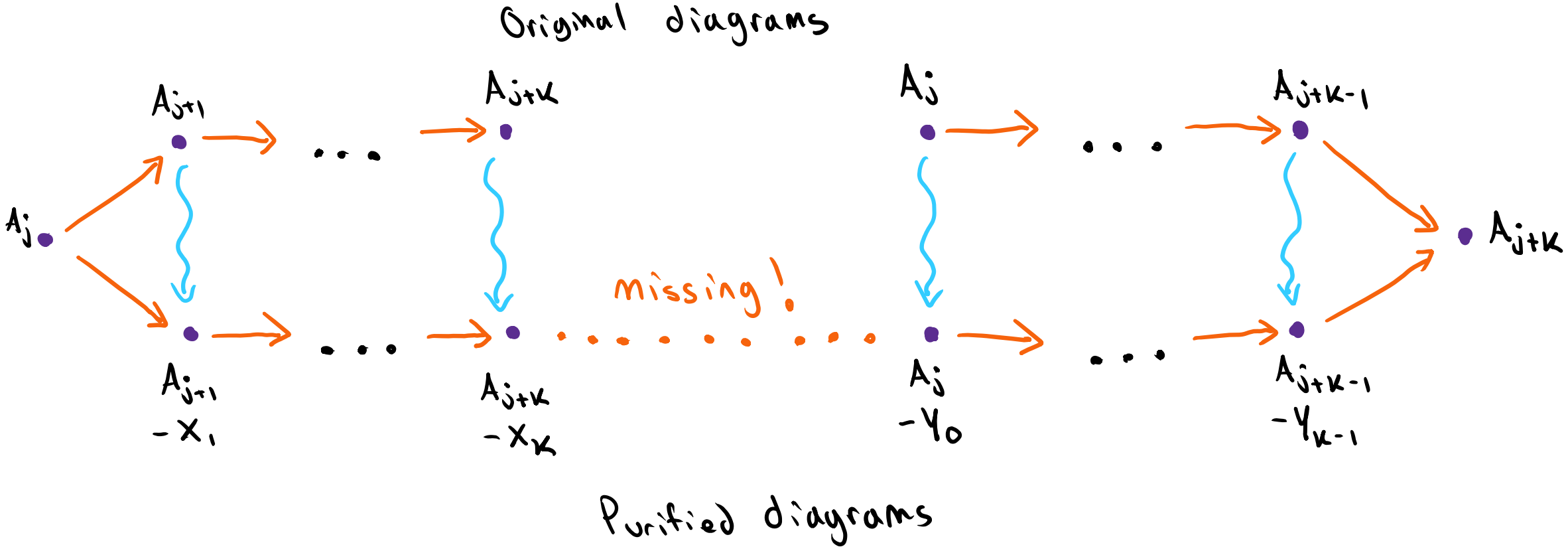
The missing ingredient is a path from $A_{j+k} - X_k$ to $A_j - Y_0$.



Solution: Build More Paths

Now we have a path $(A_j, A_{j+1} - X_1, \ldots, A_{j+k} - X_k)$ and a path $(A_j - Y_0, \ldots, A_{j+k-1} - Y_{k-1}, A_{j+k})$.

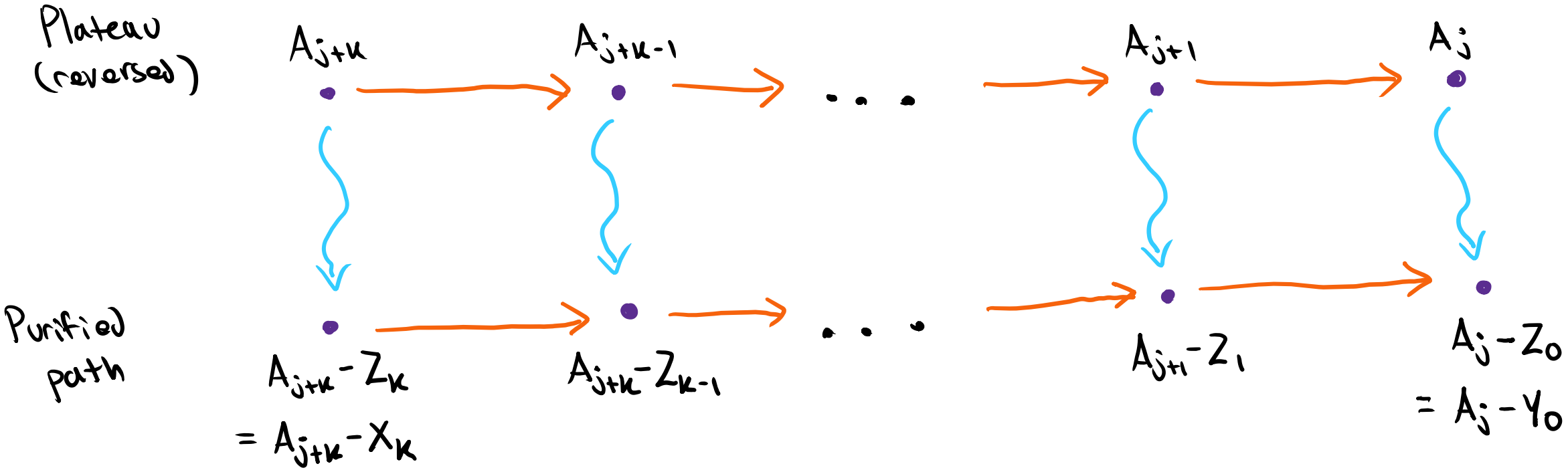
The missing ingredient is a path from $A_{j+k} - X_k$ to $A_j - Y_0$.



Solution: Build More Paths

To find such a path, we purify using the unions $Z_i = X_i \cup Y_i$ for $0 \leq i \leq k$.

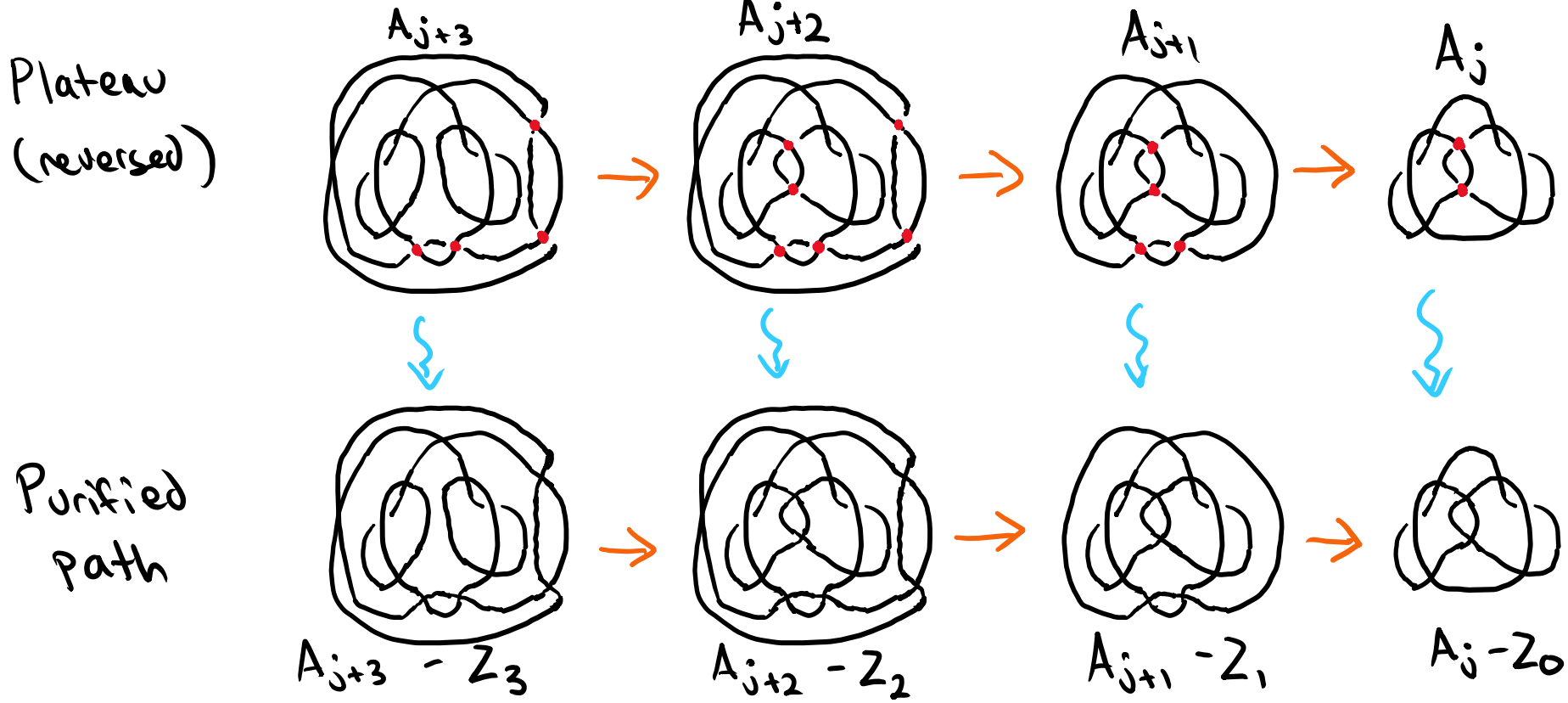
It is straightforward to check that the result of purifying using the sets Z_i is a path $(A_{j+k} - Z_k, \ldots, A_j - Z_0)$ from $A_{j+k} - X_k$ to $A_j - Y_0$.



Solution: Build More Paths

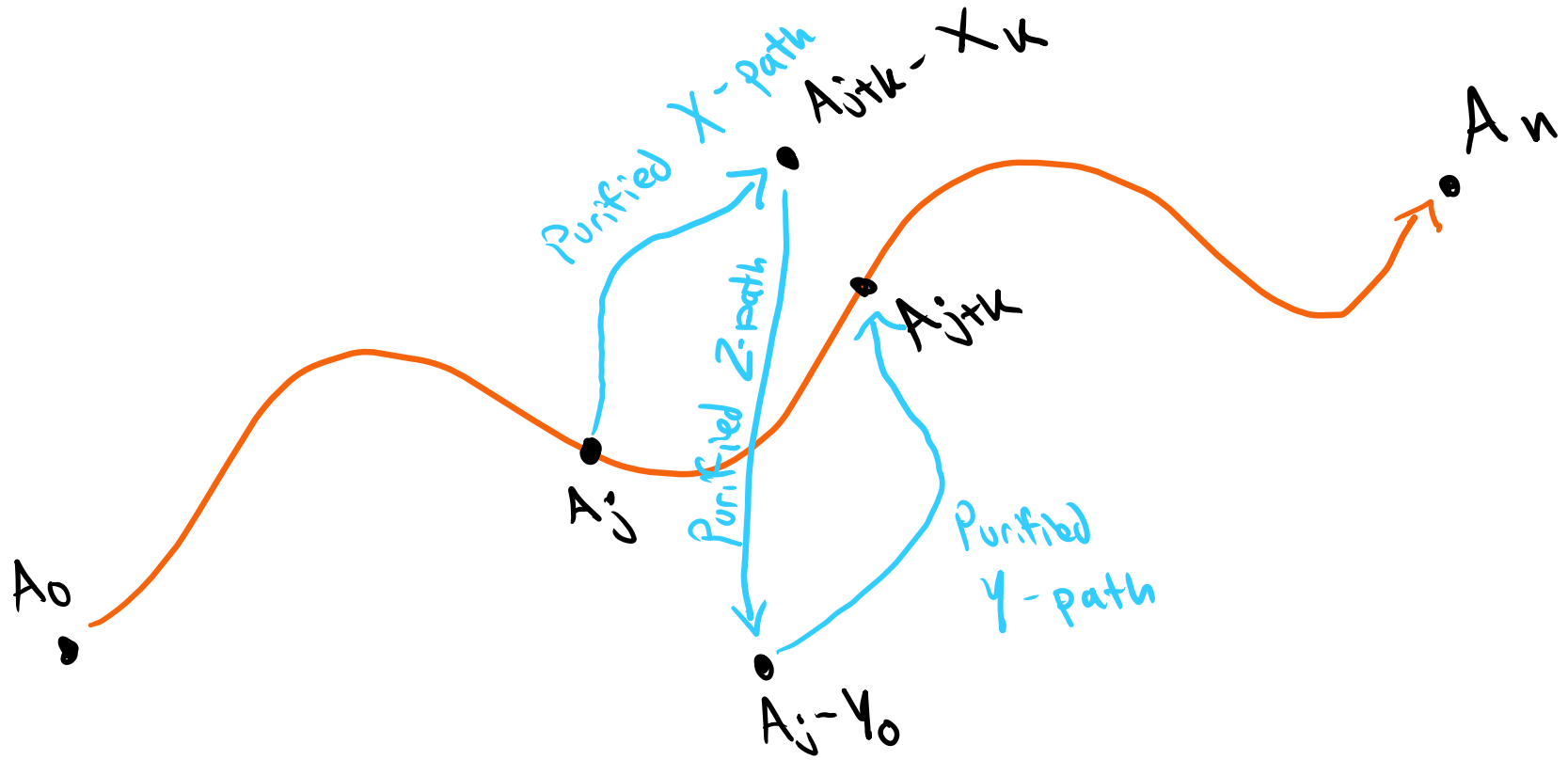
To find such a path, we purify using the unions $Z_i = X_i \cup Y_i$ for $0 \leq i \leq k$.

It is straightforward to check that the result of purifying using the sets Z_i is a path $(A_{j+k} - Z_k, \ldots, A_j - Z_0)$ from $A_{j+k} - X_k$ to $A_j - Y_0$.



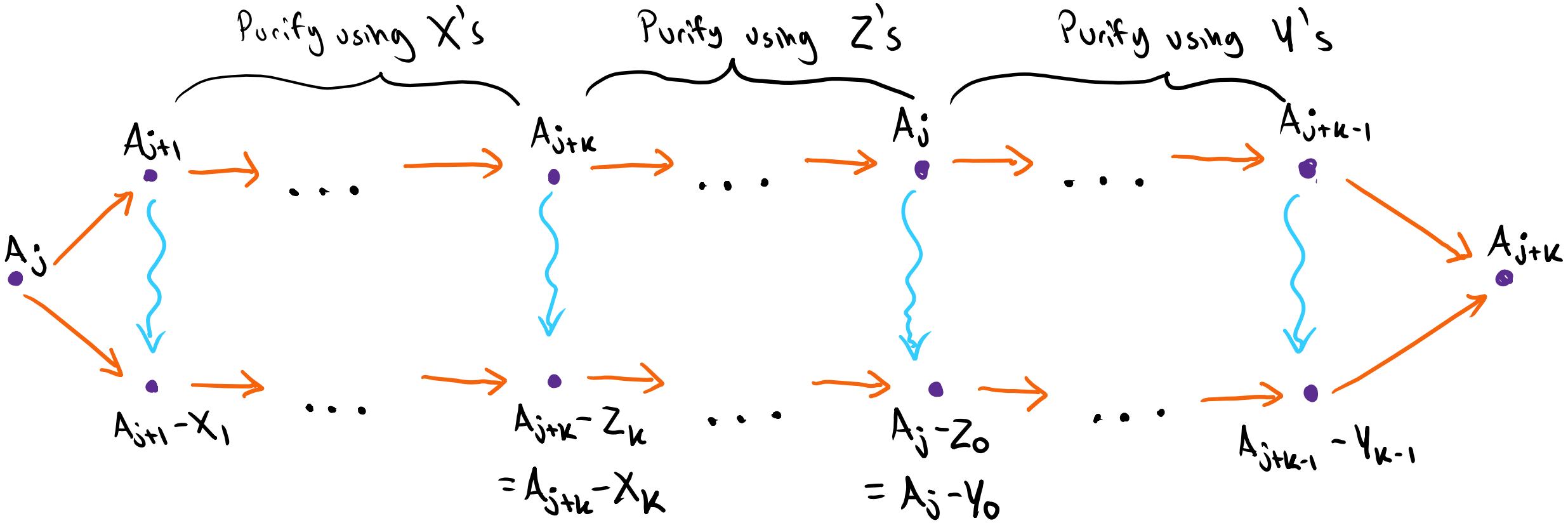
Stitching the Paths Together

We stitch all of our purified paths together to obtain an alternate path from A_j to A_{j+k}, as depicted below.

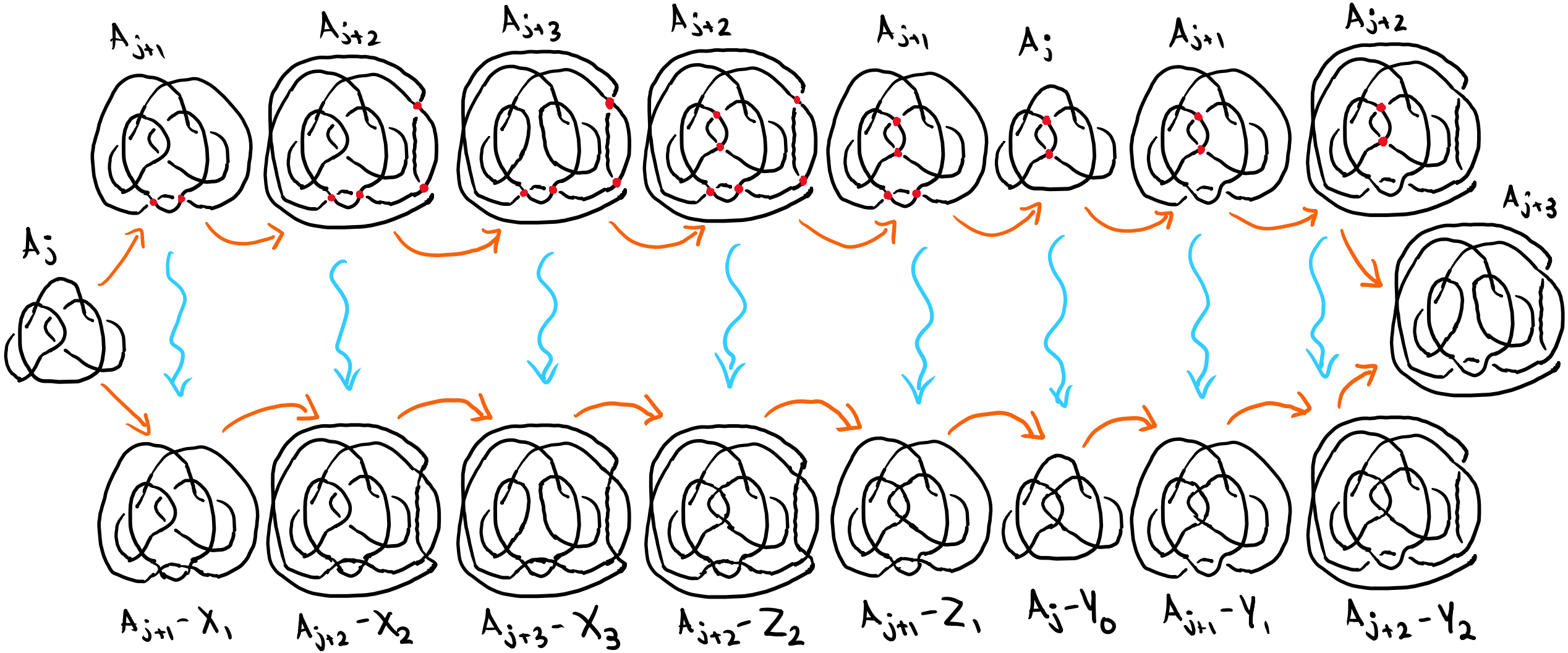


Stitching the Paths Together

We stitch all of our purified paths together to obtain an alternate path from A_j to A_{j+k}, as depicted below.



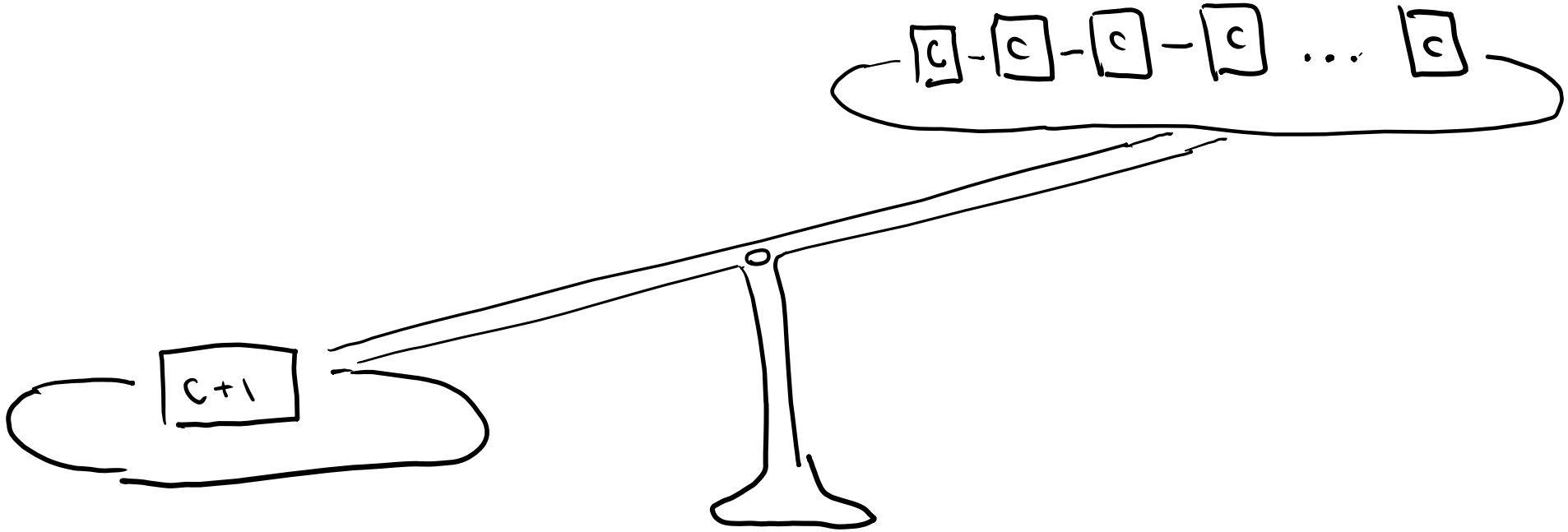
Stitching the Paths Together



Will this Terminate?

We have replaced our plateau $(A_j, \dots, A_{j+k})$ in $(A_0, \dots, A_n)$ with an alternative path which is three times as long. Why should this process terminate?

The answer is that if we consider a weighing system (i.e. total ordering) on paths in which a diagram of with $c+1$ crossings weighs more than any number of diagrams with c crossings, then performing our replacement indeed reduces the weight of $(A_0, \dots, A_n)$.



Non-Emptiness of the Sets

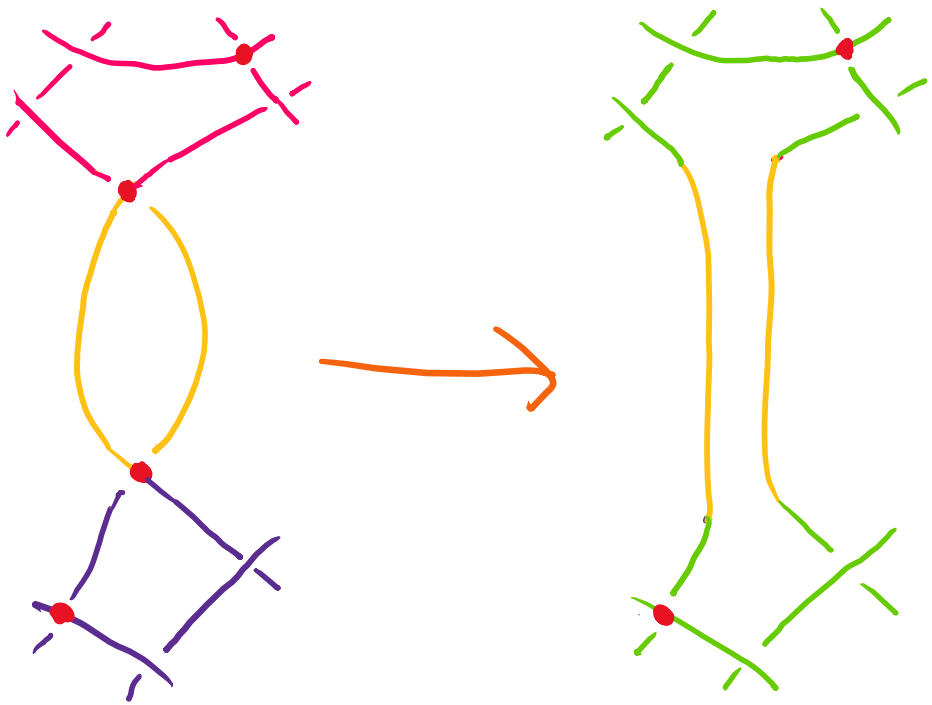
Technically, for our replacement to reduce the weight of $(A_0, \dots, A_n)$, we also need that the bad sets $X_1, \dots, X_{k-1}$ and $Y_1, \dots, Y_{k-1}$ are all non-empty.

Since X_1 and Y_{k-1} were chosen to be non-empty, it suffices to check that non-emptiness is a property preserved while choosing our even-bounding bad sets.

Non-Emptiness of the Sets

Since we followed rule 1 in our constructions, the only way non-emptiness could potentially not be preserved is if all bad crossings are destroyed by a backwards VR-II move. As shown below, the move would have to be non-local.

Only the initial and final moves involved in our plateau are non-local, so it follows that the sets $X_1, \dots, X_{k-1}$ and $Y_1, \dots, Y_{k-1}$ are indeed non-empty.



If the move were local, the pink and purple cycles would both have to contain at least one other bad crossing in order to contain an even number.

However, then not every bad crossing would be destroyed by the move.

Thank you for listening!