Graph Theory

Part Two

Outline for Today

- ***Walks, Paths, and Reachability***
  - Walking around a graph.
- ***Graph Complements***
  - Flipping what's in a graph.
- ***The Teleported Train Problem***
  - A very exciting commute.
- ***The CBS Theorem***
  - Cardinality meets graph theory!

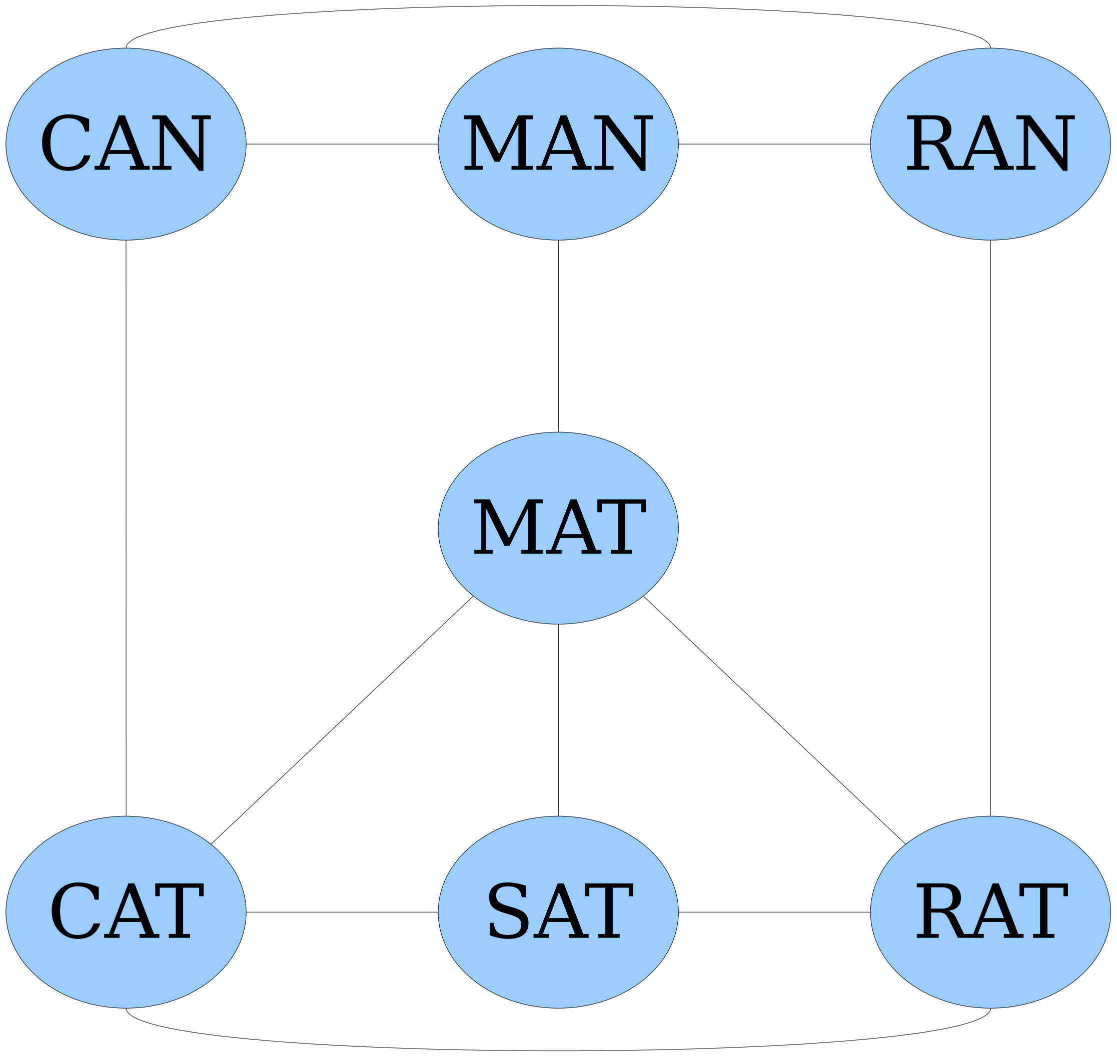
Recap from Last Time

Graphs and Digraphs

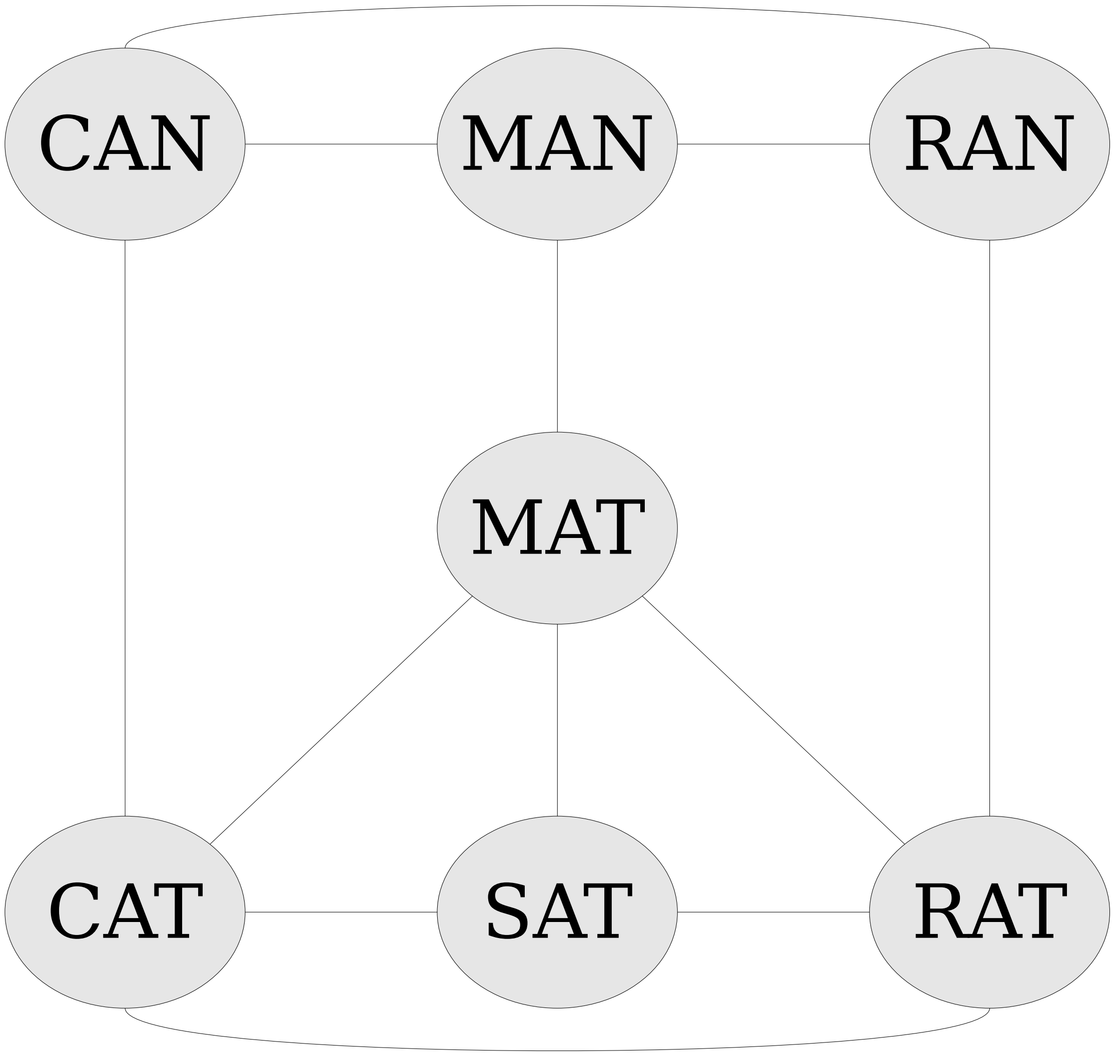
- A ***graph*** is a pair $G = (V, E)$ of a set of nodes V and set of edges E.
  - Nodes can be anything.
  - Edges are ***unordered pairs*** of nodes. For example, if $\{u, v\} \in E$, then there's an edge from u to v.
- A ***digraph*** is a pair $G = (V, E)$ of a set of nodes V and set of directed edges E.
  - Each edge is represented as the ordered pair (u, v) indicating an edge from u to v.

New Stuff!

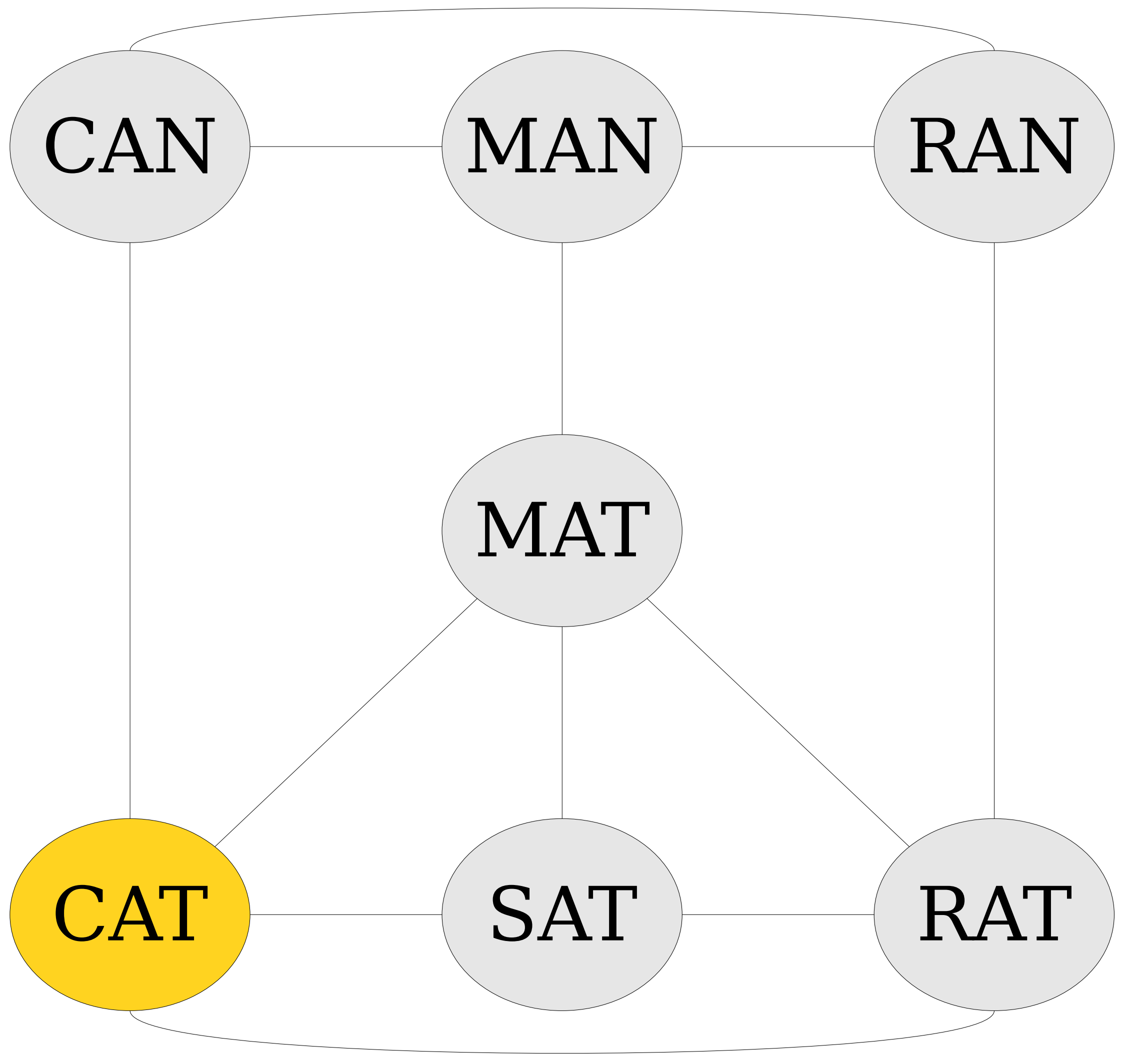
Walks, Paths, and Reachability



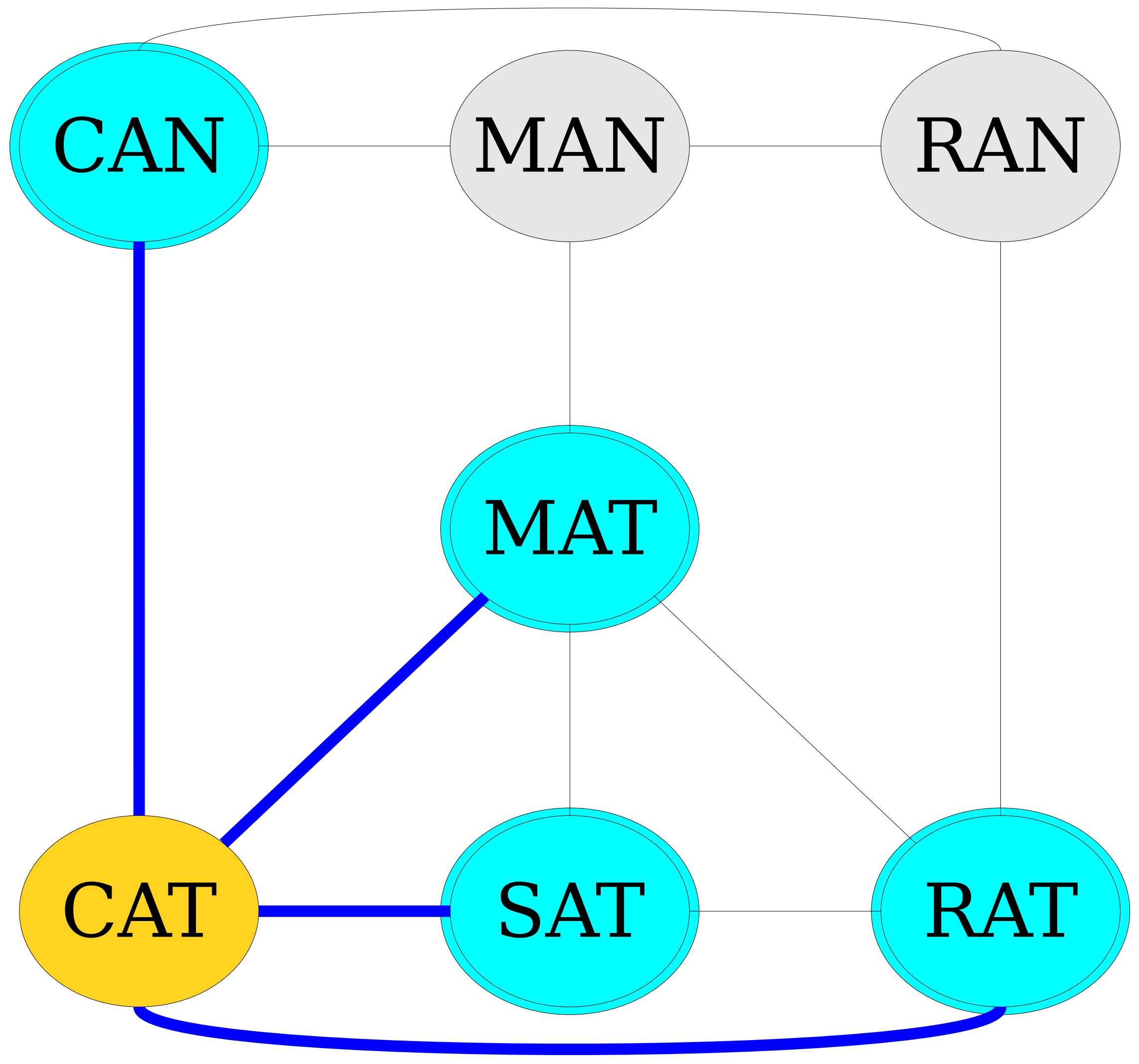
Two nodes are called ***adjacent*** if there is an edge between them.



Two nodes are called ***adjacent*** if there is an edge between them.



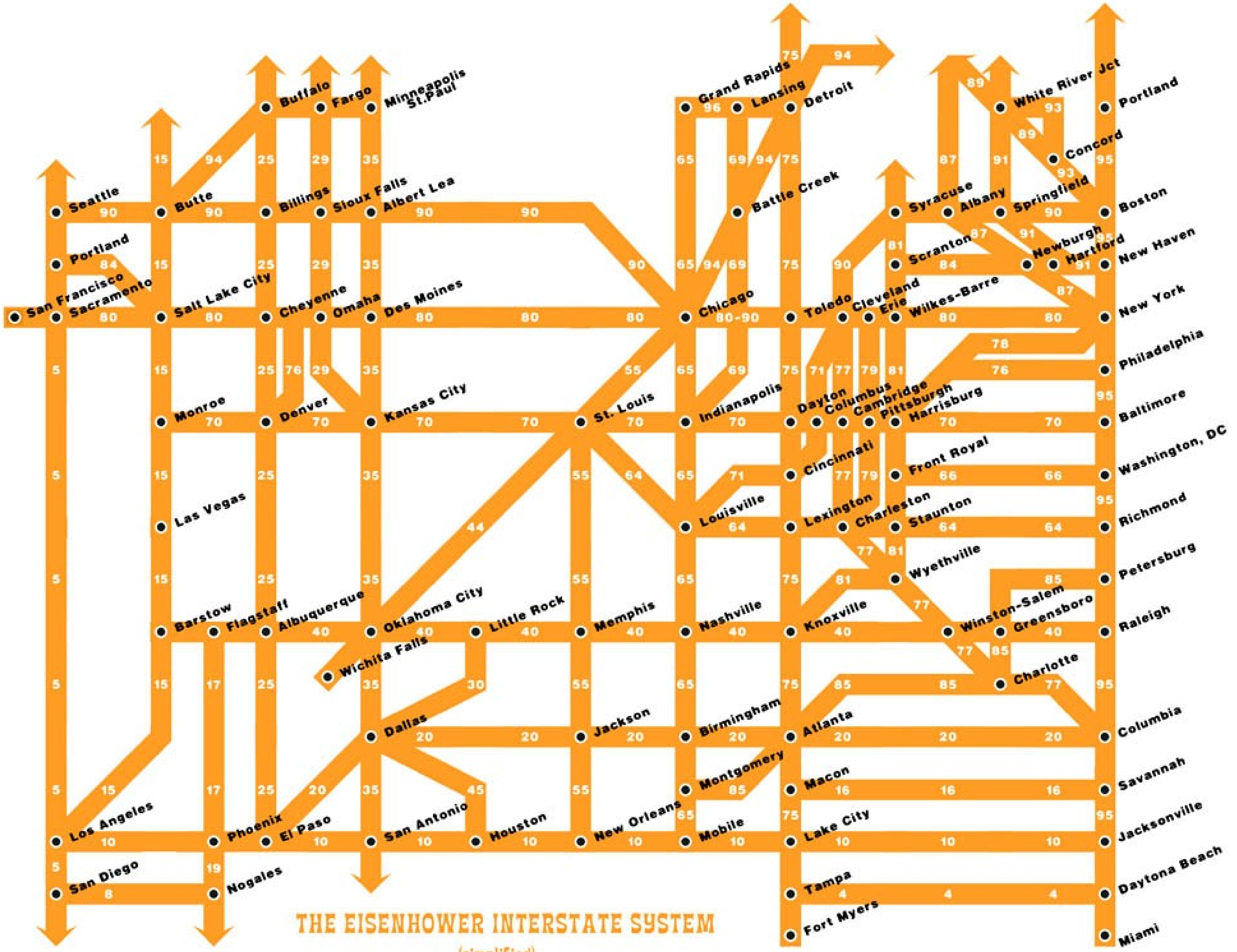
Two nodes are called ***adjacent*** if there is an edge between them.



Two nodes are called ***adjacent*** if there is an edge between them.

Using our Formalisms

- Let $G = (V, E)$ be an (undirected) graph.
- Intuitively, two nodes are adjacent if they're linked by an edge.
- Formally speaking, we say that two nodes $u, v \in V$ are ***adjacent*** if we have $\{u, v\} \in E$.
- There isn’t an analogous notion for directed graphs. We usually just say “there’s an edge from u to v” as a way of reading $(u, v) \in E$ aloud.

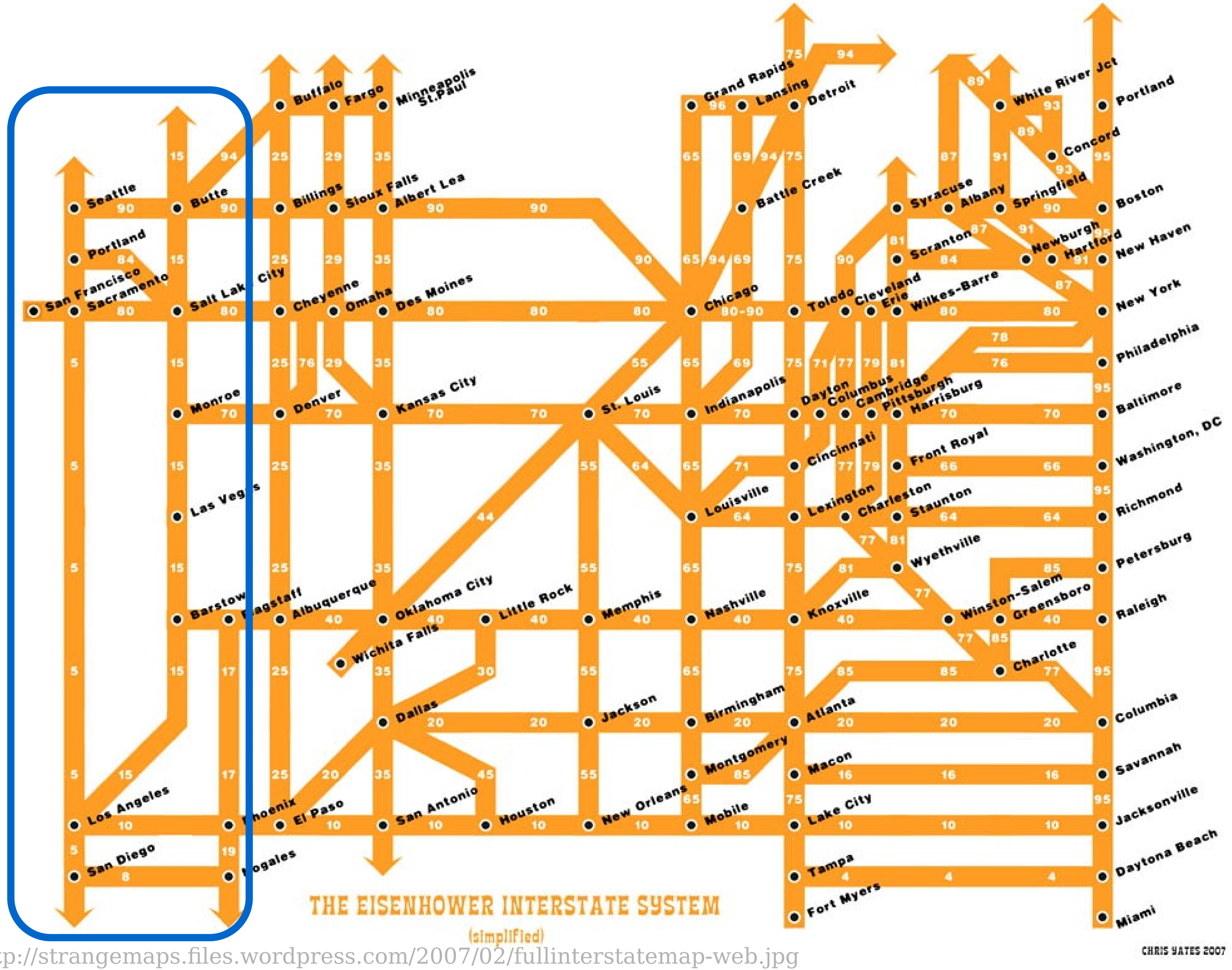


THE EISENHOWER INTERSTATE SYSTEM

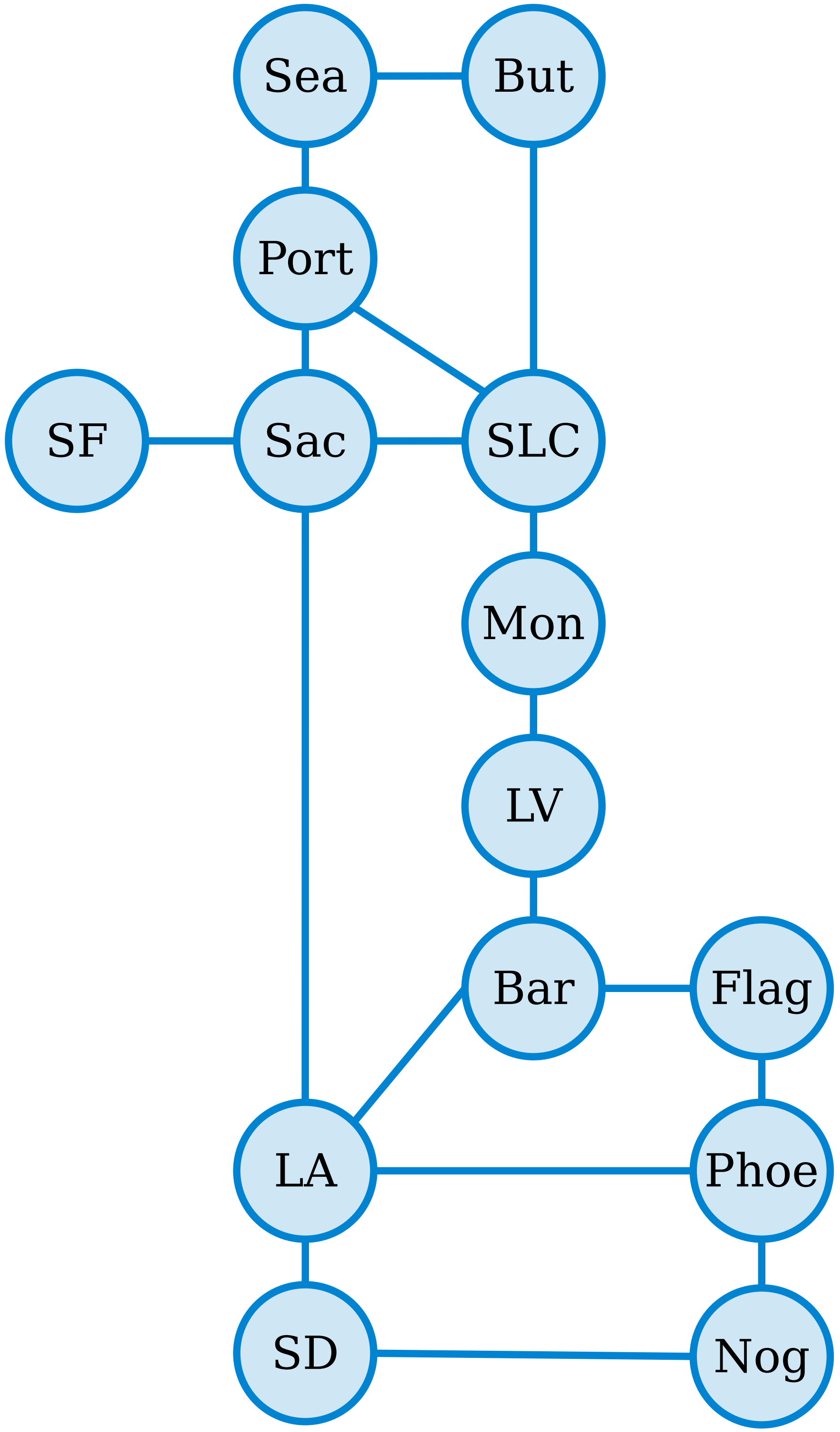
(simplified)

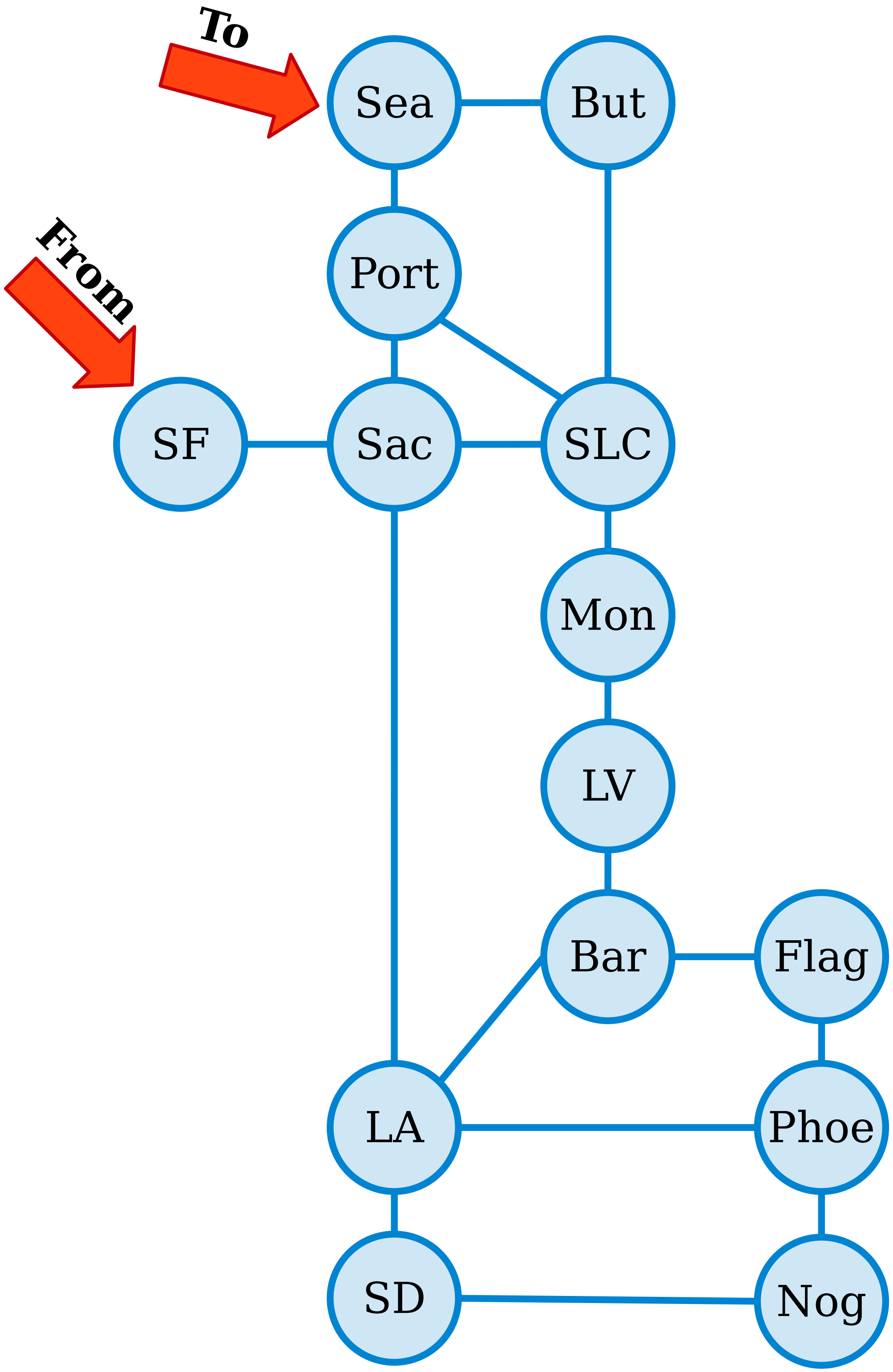
CHRIS YATES 2007

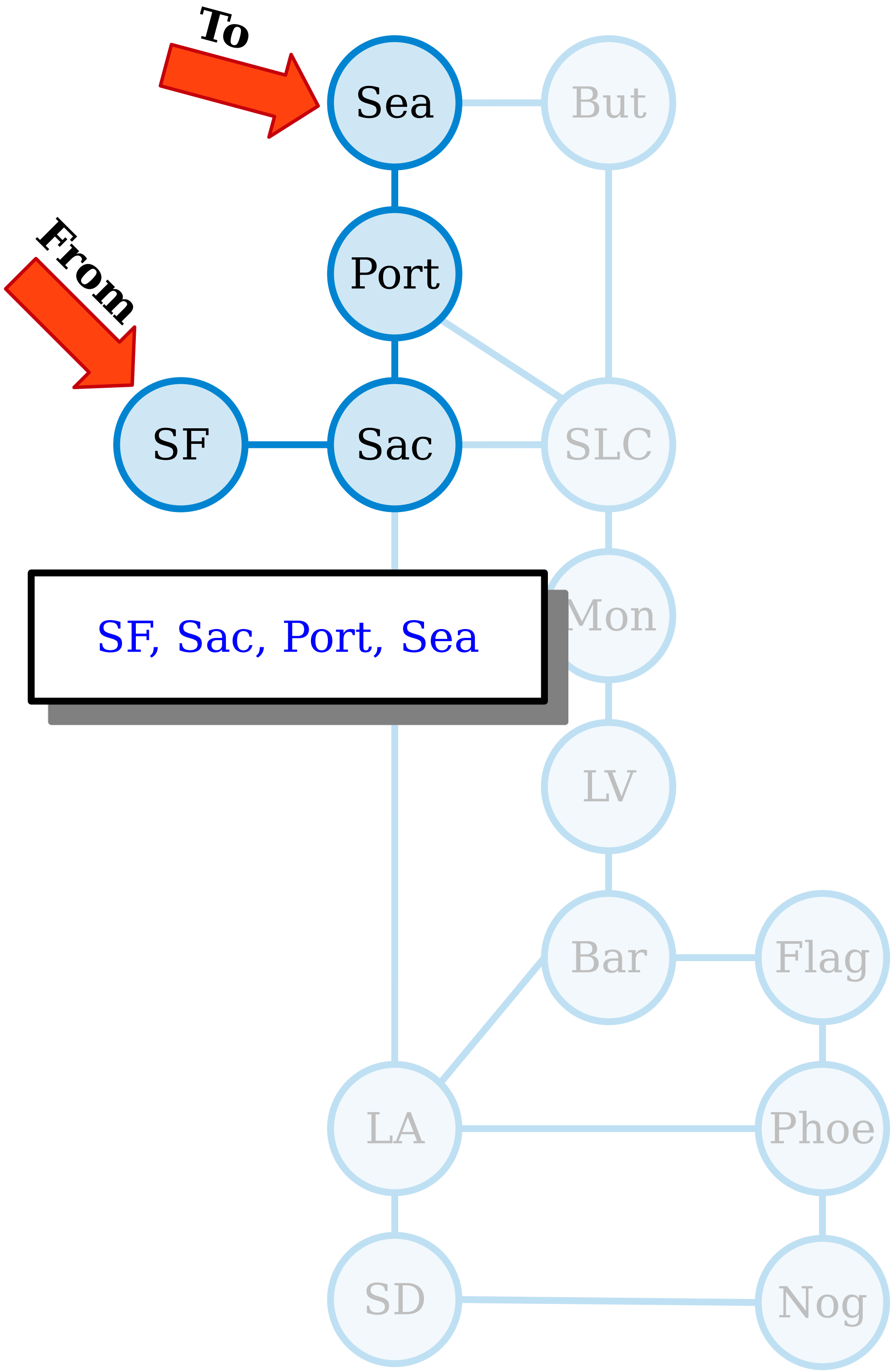
http://strangemaps.files.wordpress.com/2007/02/fullinterstatemap-web.jpg



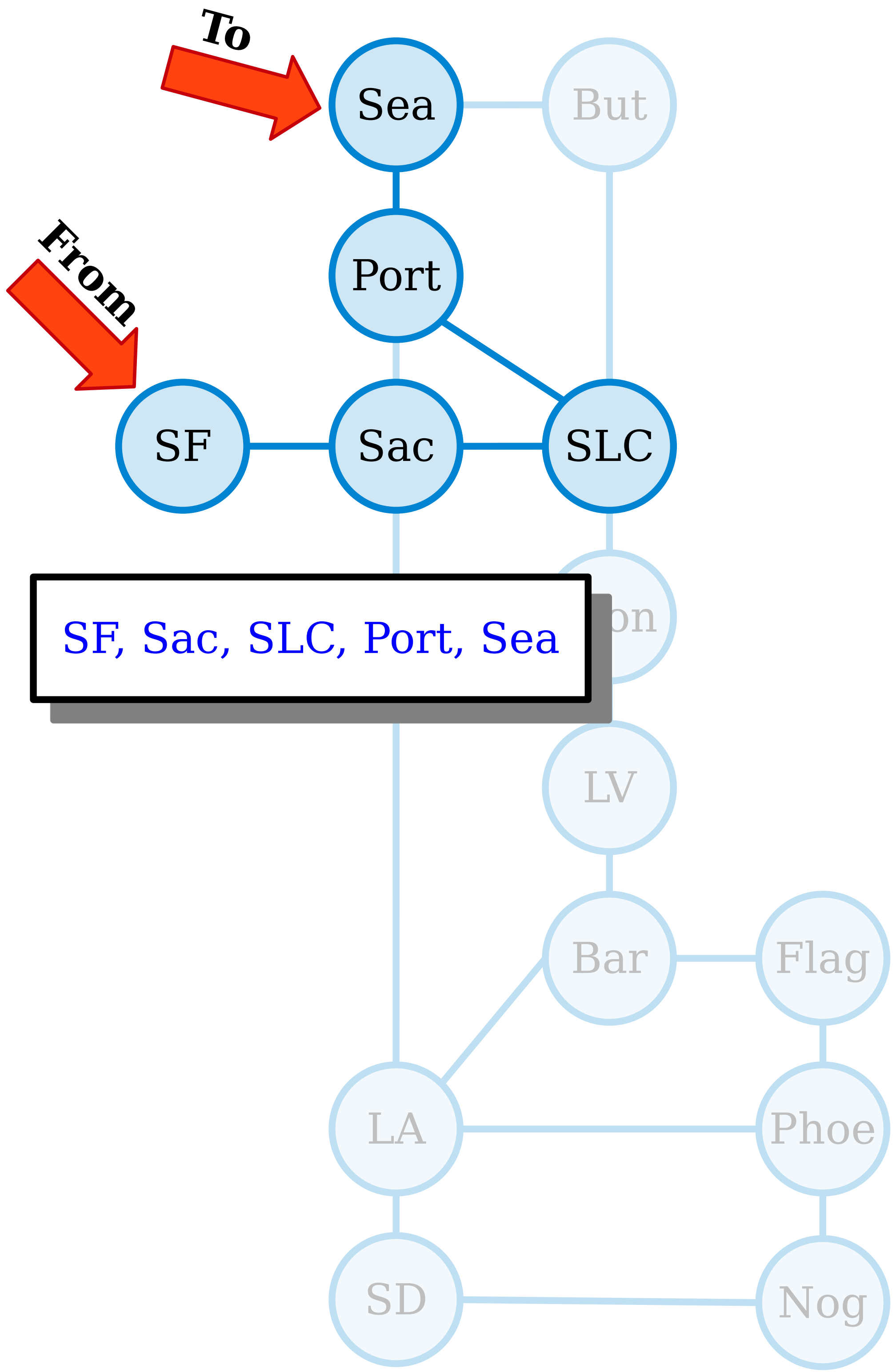
http://strangemaps.files.wordpress.com/2007/02/fullinterstatemap-web.jpg


Sea
But
Port
SF
Sac
SLC
Mon
LV
Bar
Flag
LA
Phoe
SD
Nog

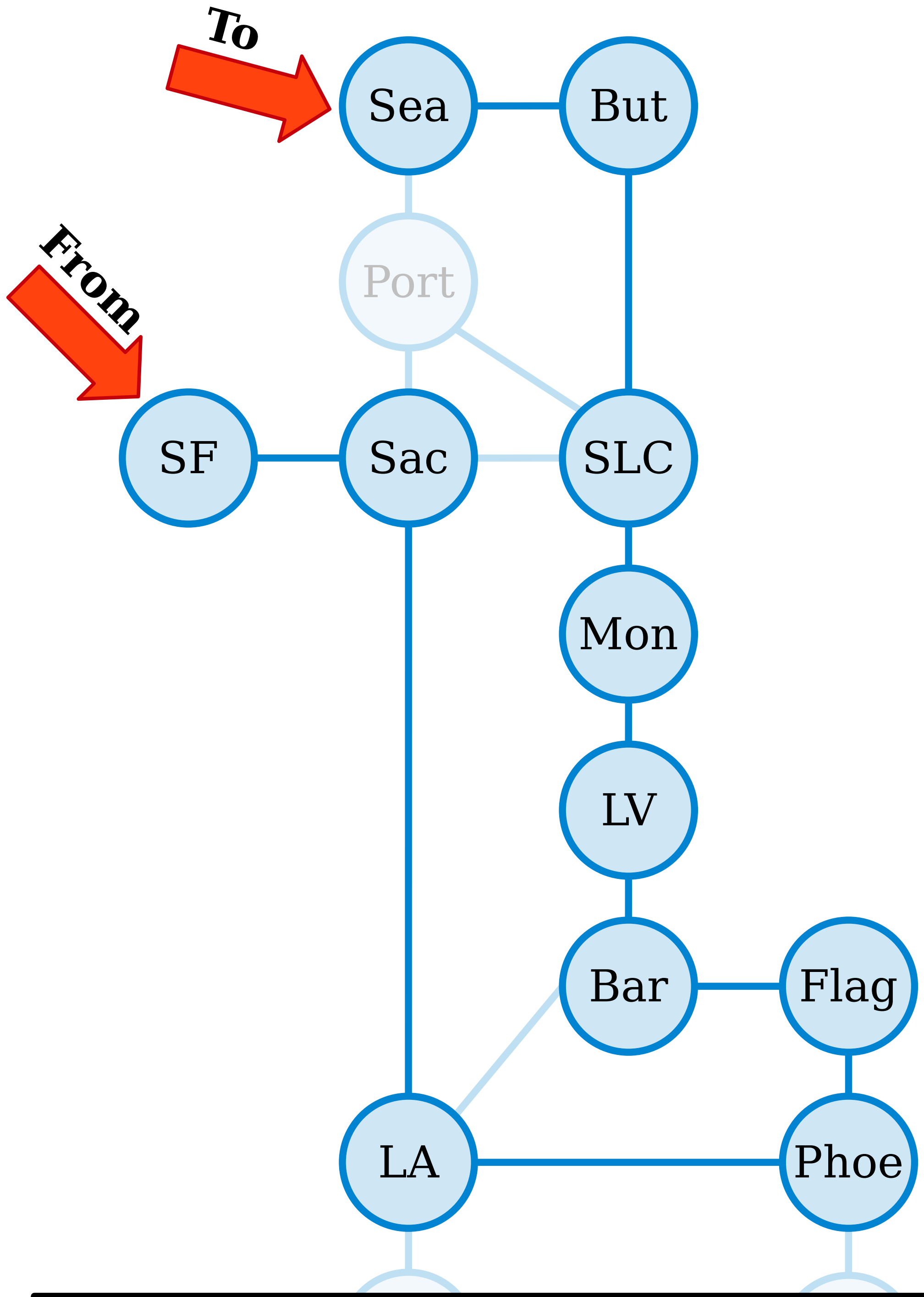

To
From
Sea
But
Port
SF
Sac
SLC
Mon
LV
Bar
Flag
LA
Phoe
SD
Nog



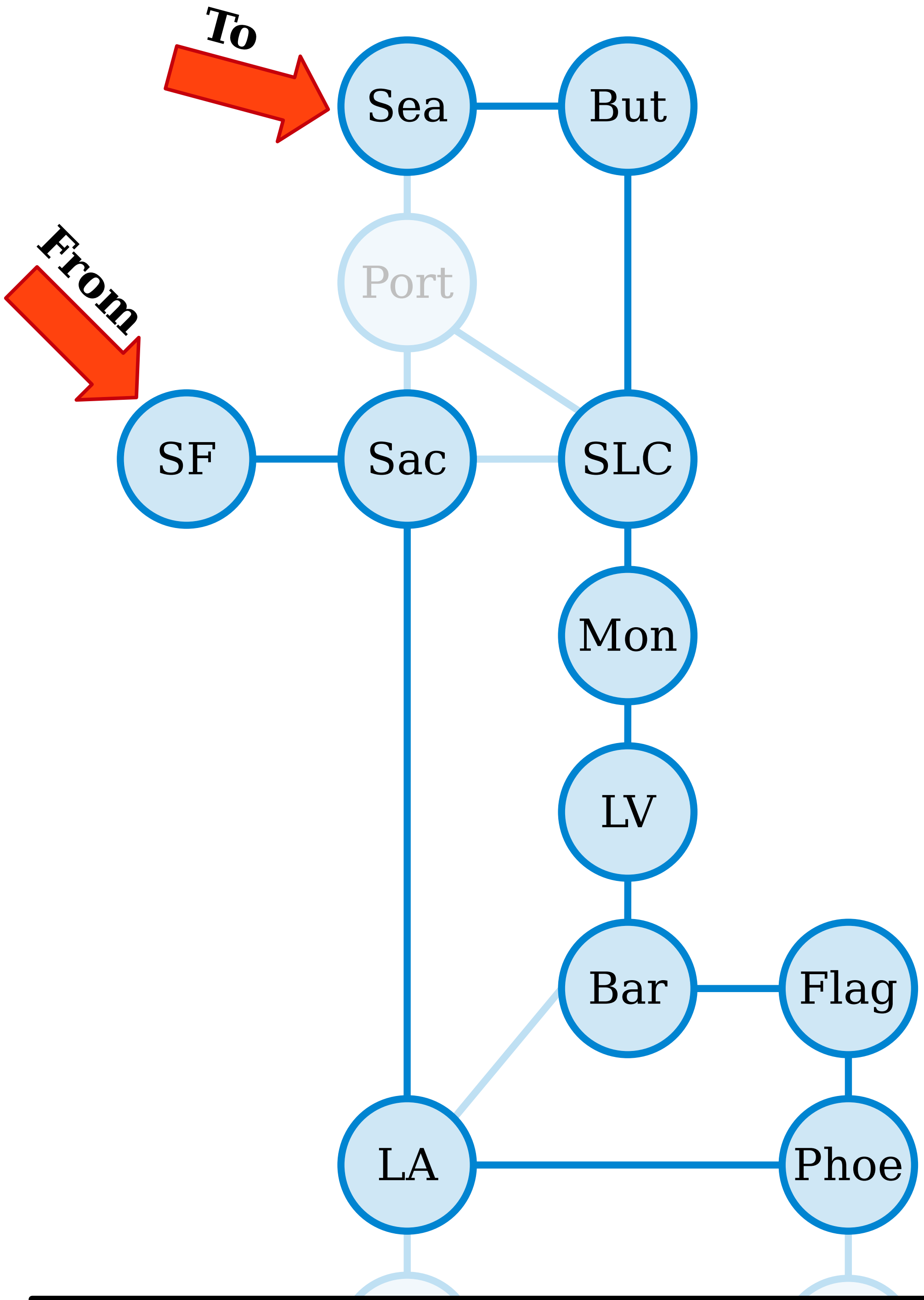
SF, Sac, Port, Sea



SF, Sac, SLC, Port, Sea

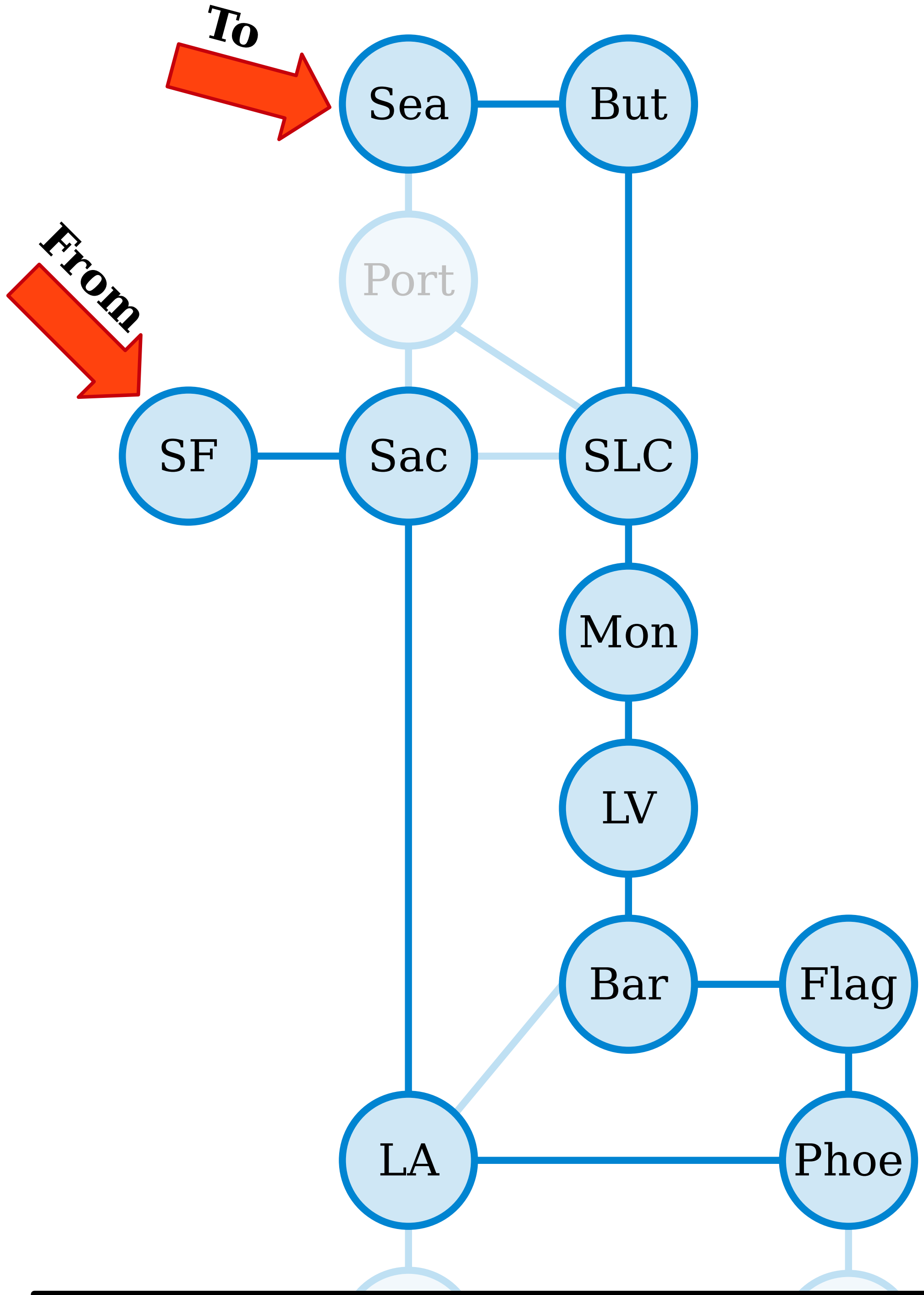


SF, Sac, LA, Phoe, Flag, Bar, LV, Mon, SLC, But, Sea



A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

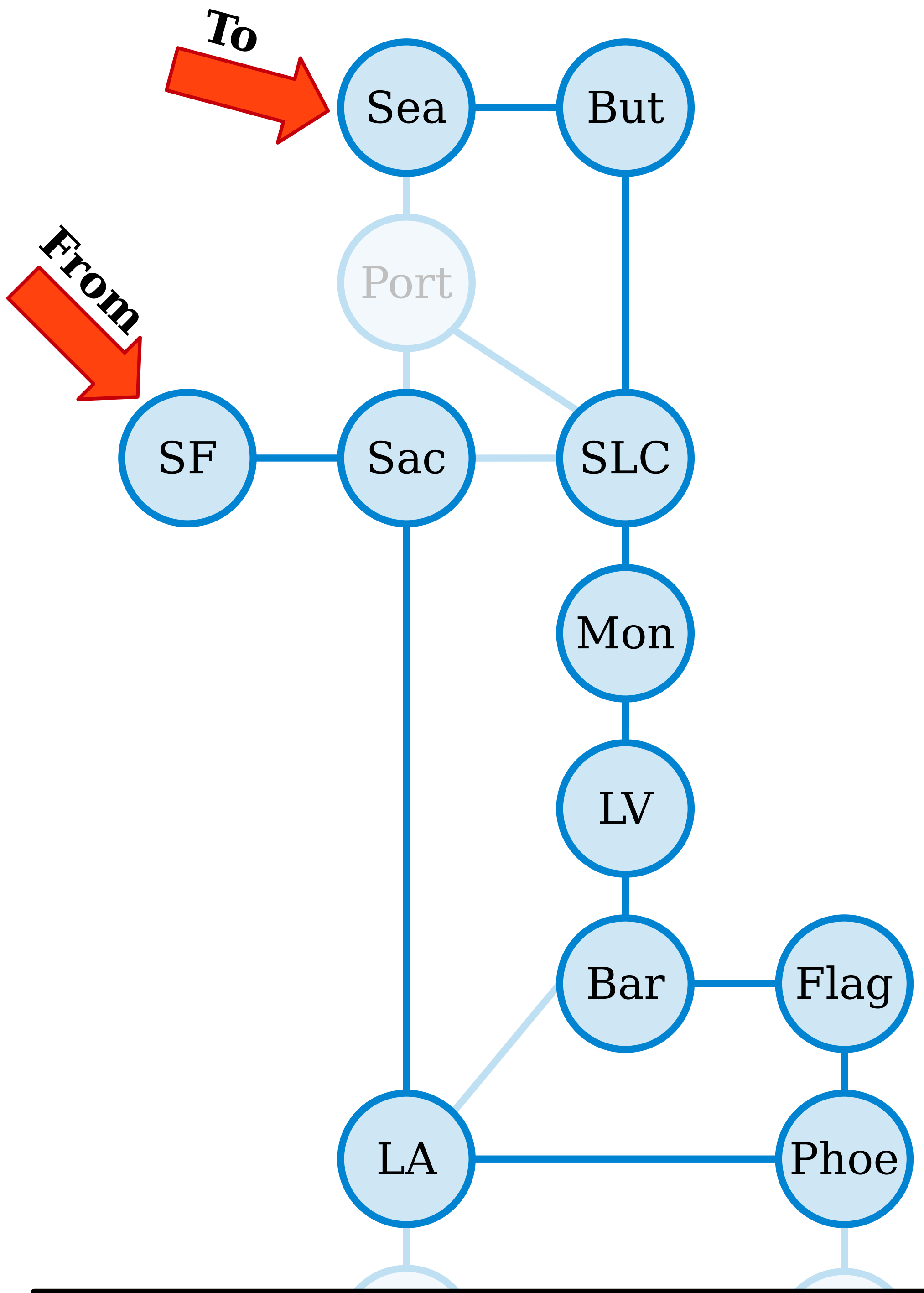
SF, Sac, LA, Phoe, Flag, Bar, LV, Mon, SLC, But, Sea



A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

SF, Sac, LA, Phoe, Flag, Bar, LV, Mon, SLC, But, Sea

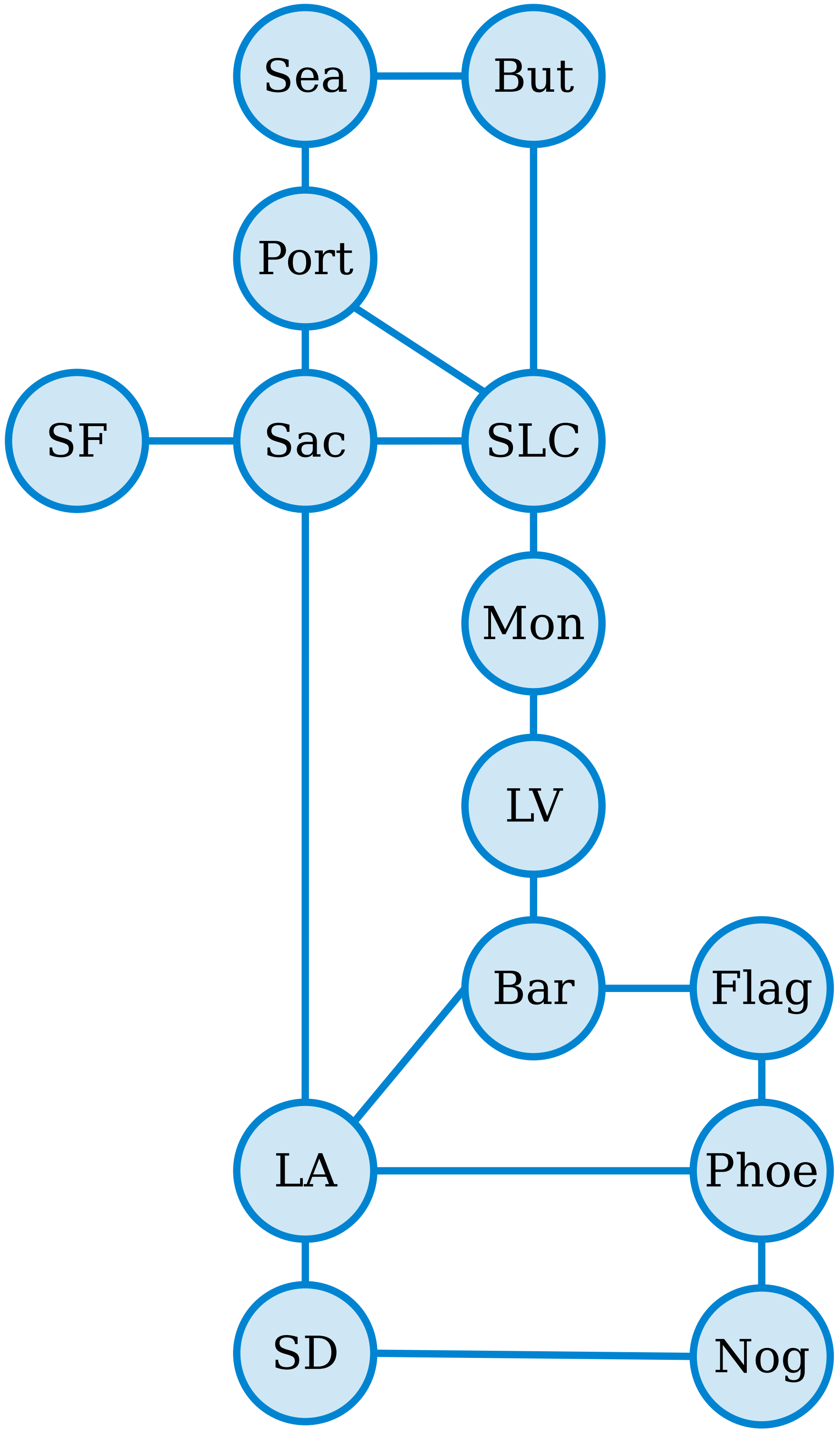


A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

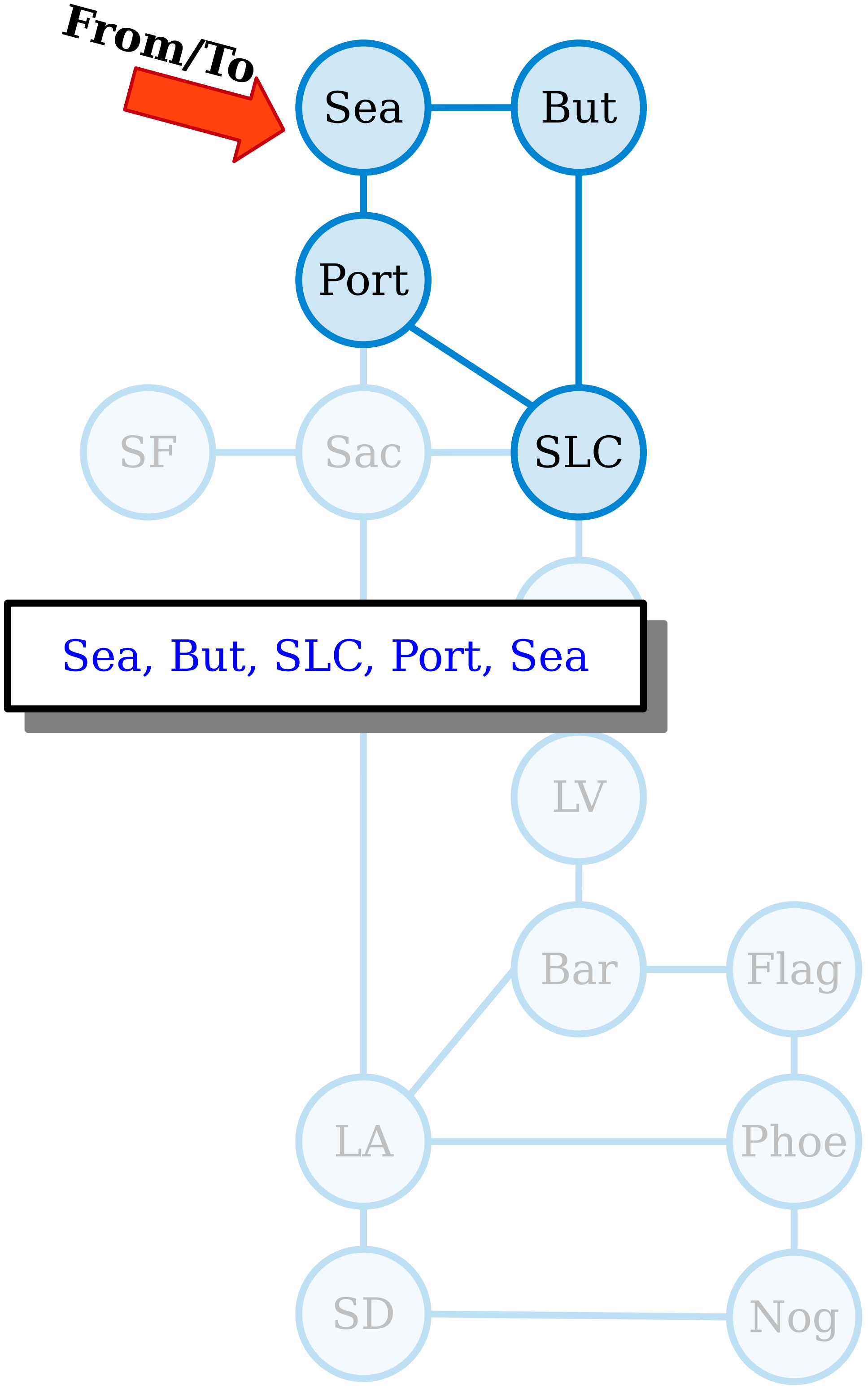
(This walk has length 10, but visits 11 cities.)

SF, Sac, LA, Phoe, Flag, Bar, LV, Mon, SLC, But, Sea



A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

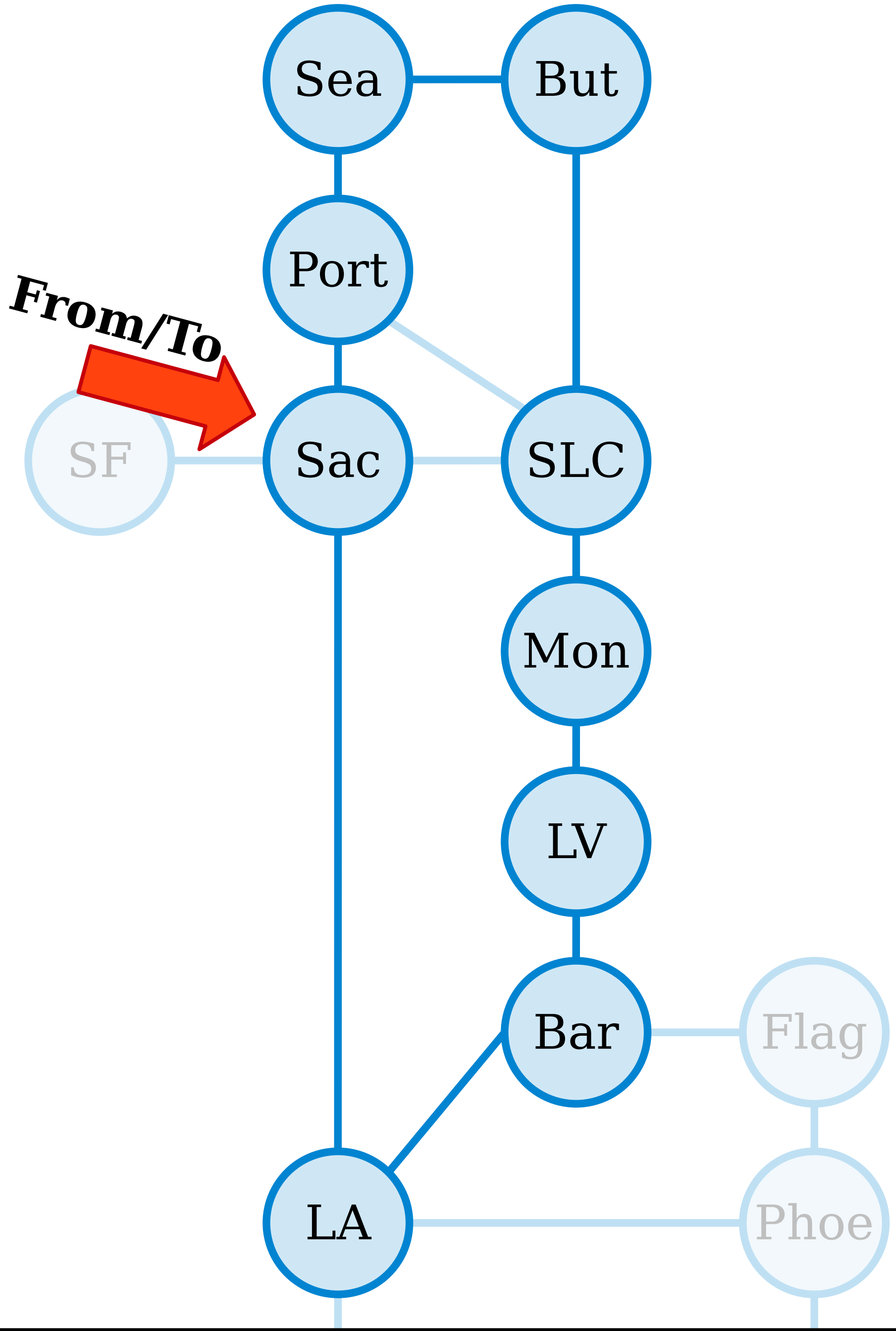
The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.



Sea, But, SLC, Port, Sea

A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

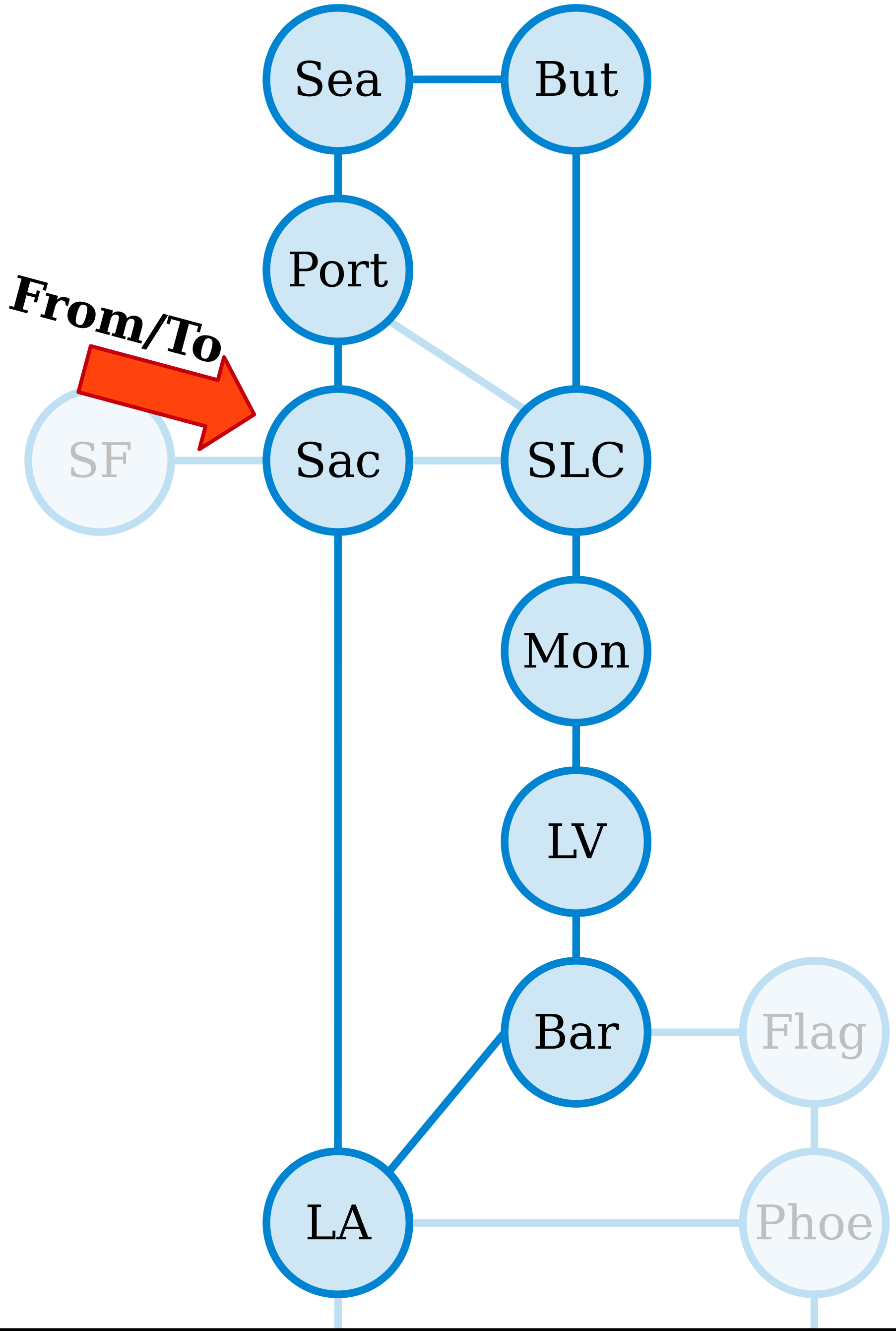
The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.



A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

Sac, Port, Sea, But, SLC, Mon, LV, Bar, LA, Sac

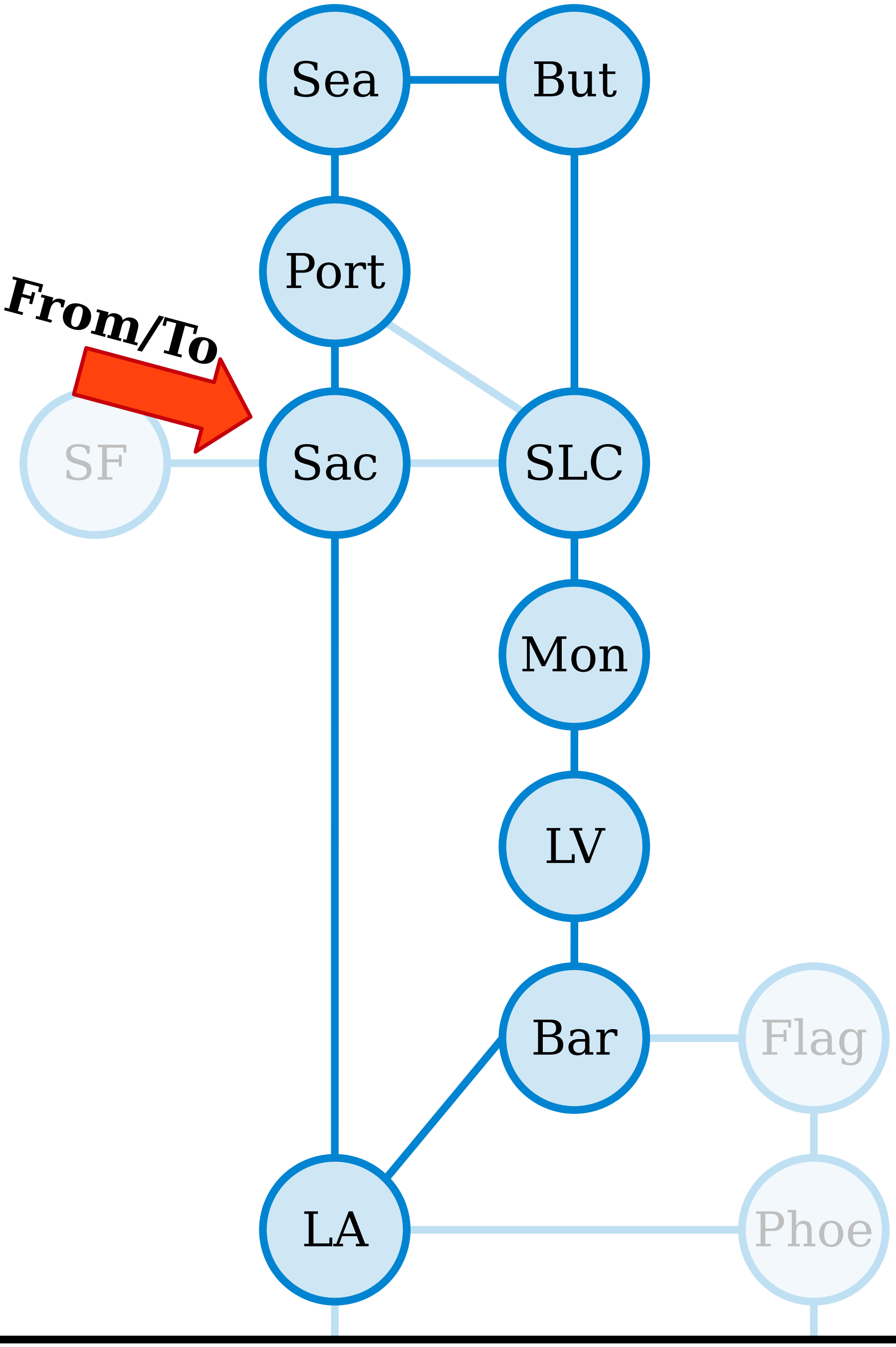


A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

Sac, Port, Sea, But, SLC, Mon, LV, Bar, LA, Sac



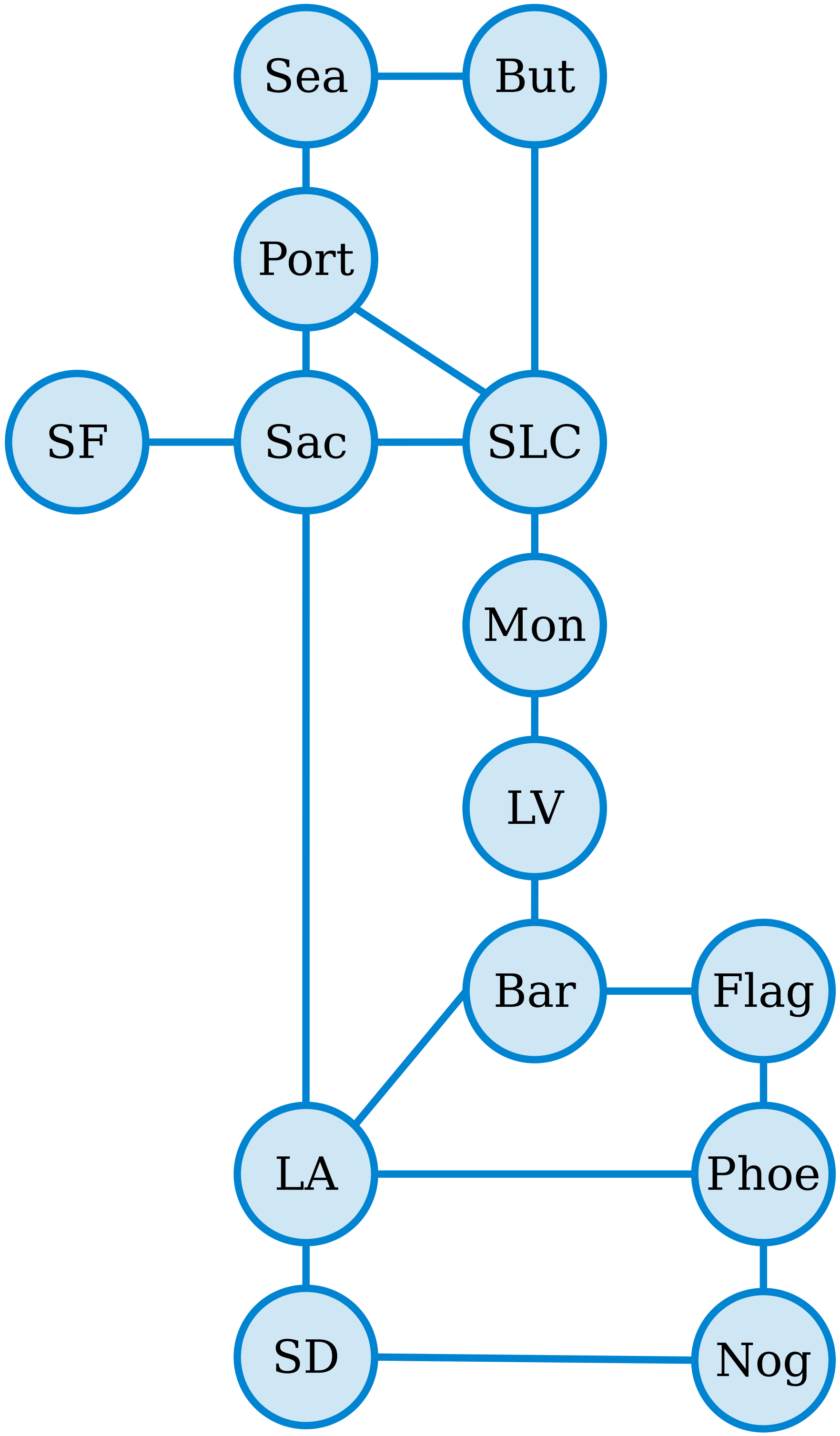
A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

(This closed walk has length nine and visits nine different cities.)

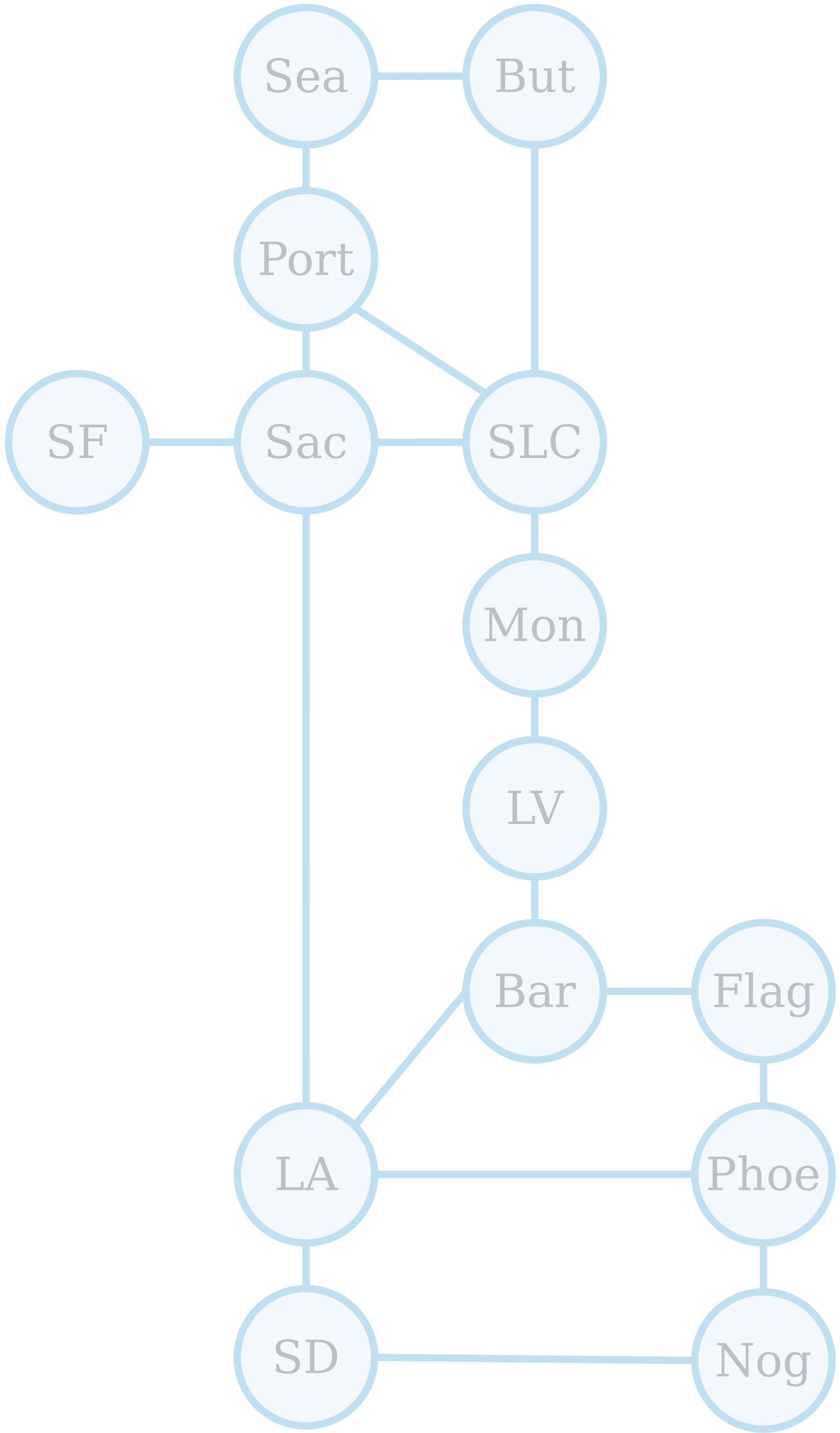
Sac, Port, Sea, But, SLC, Mon, LV, Bar, LA, Sac



A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

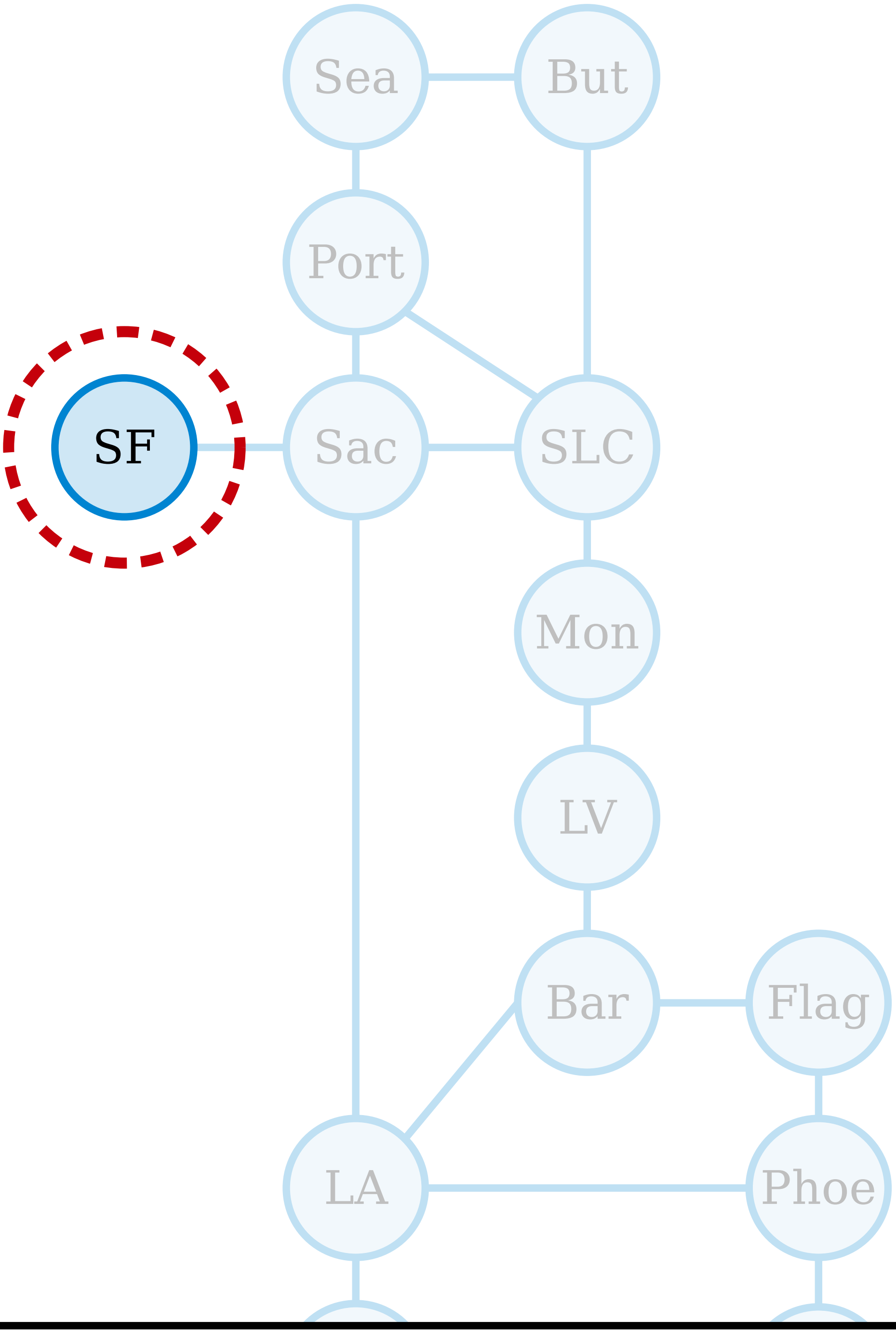
A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)



A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

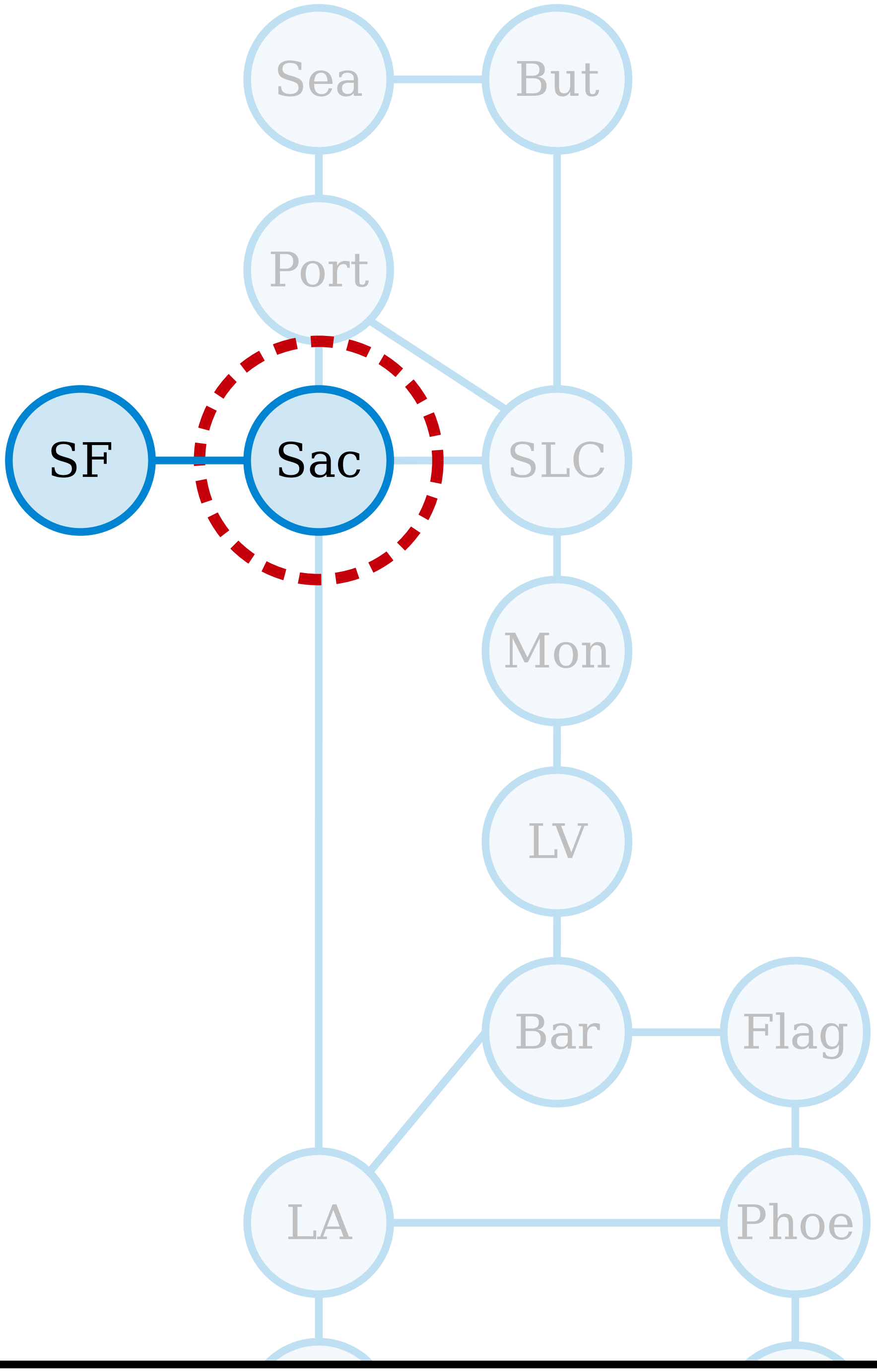


A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

SF

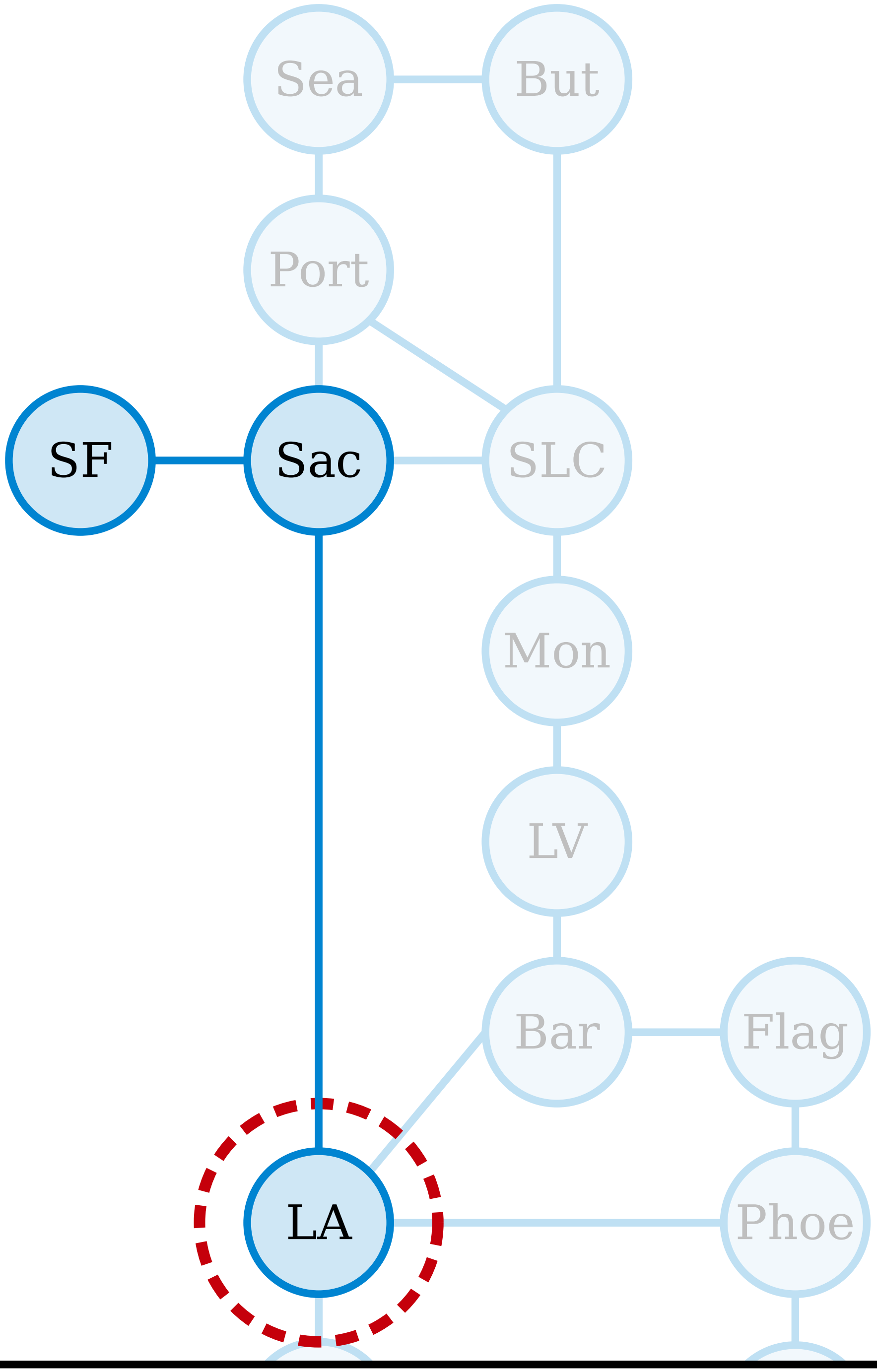


A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

SF, Sac

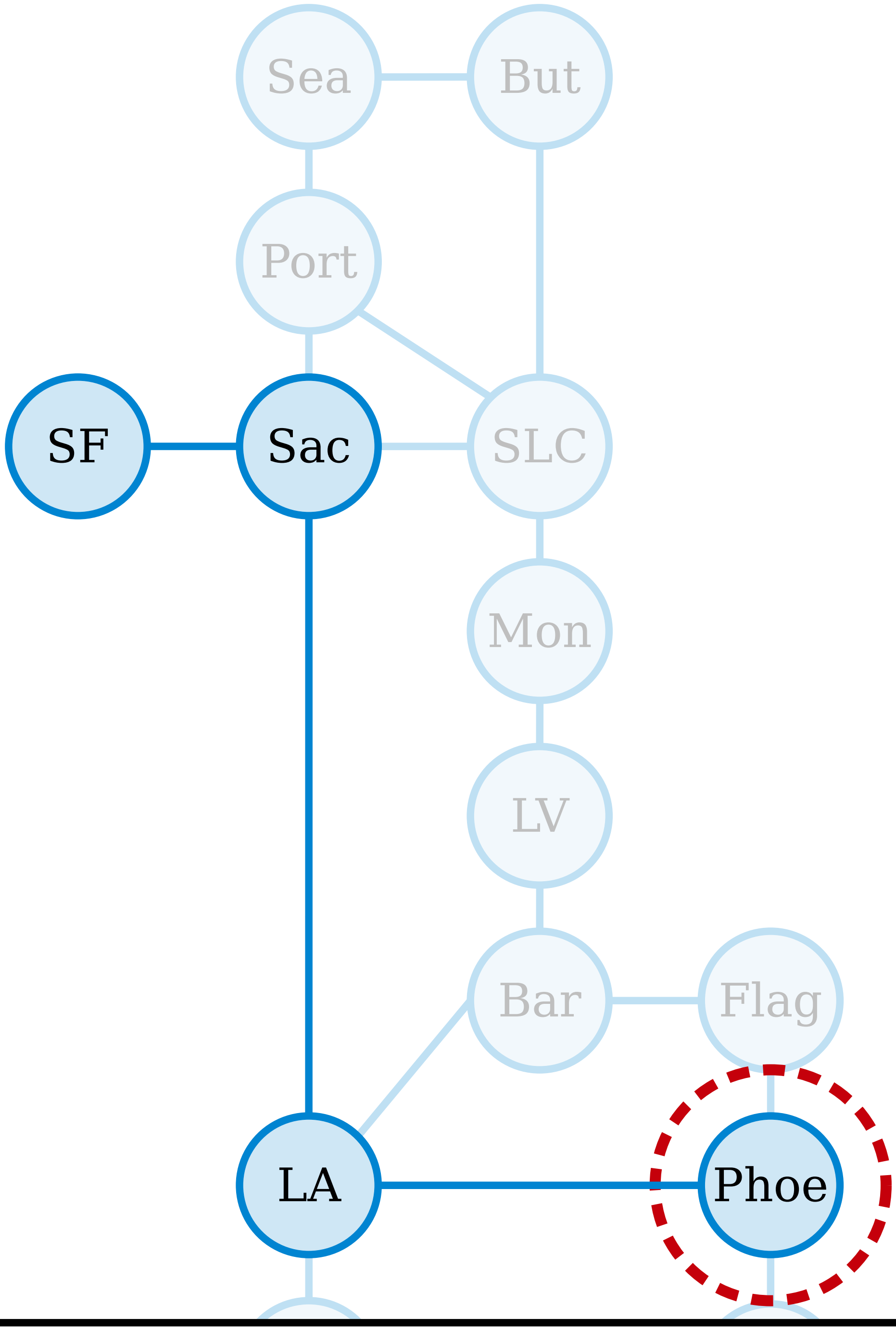


A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

SF, Sac, LA

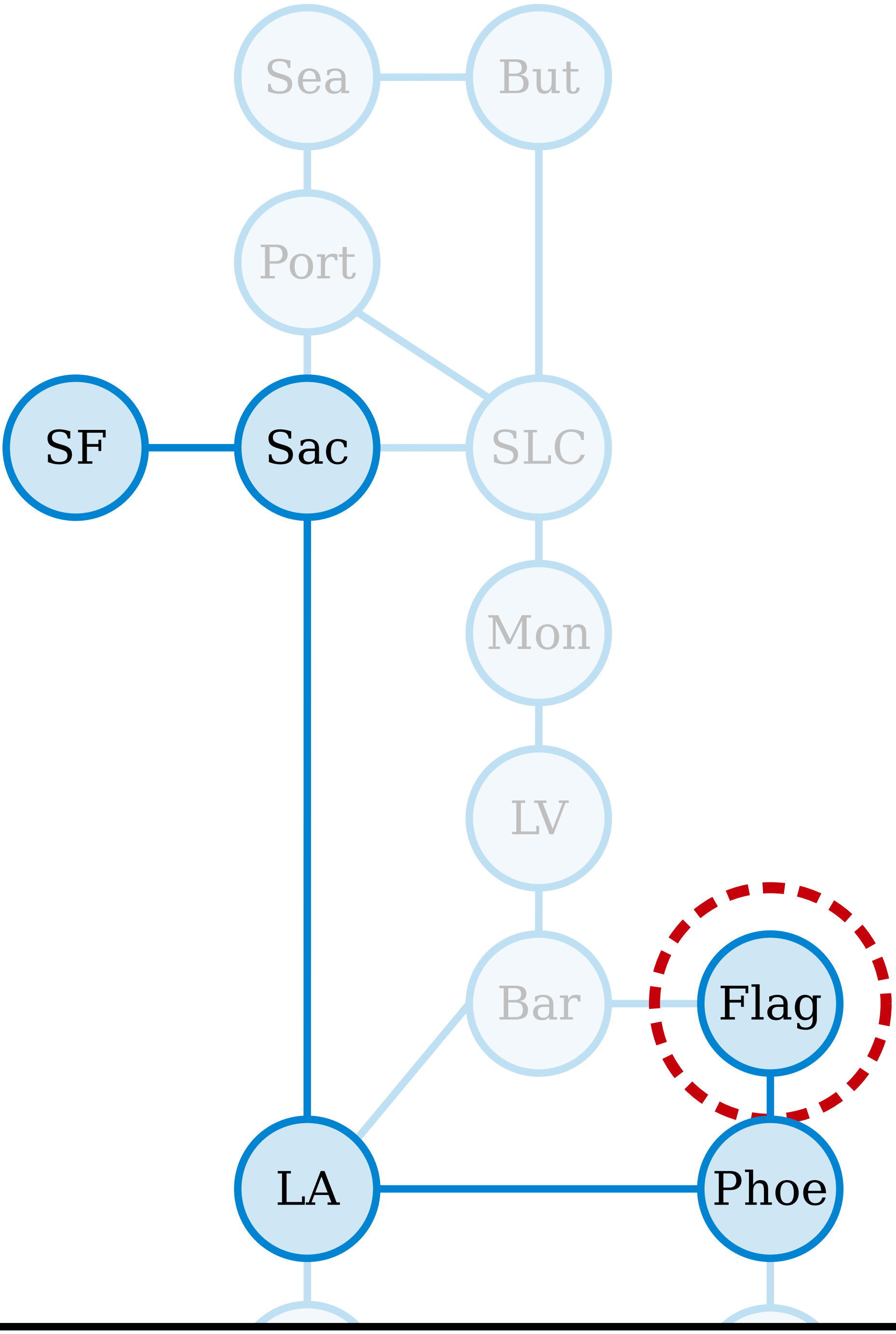


A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

SF, Sac, LA, Phoe

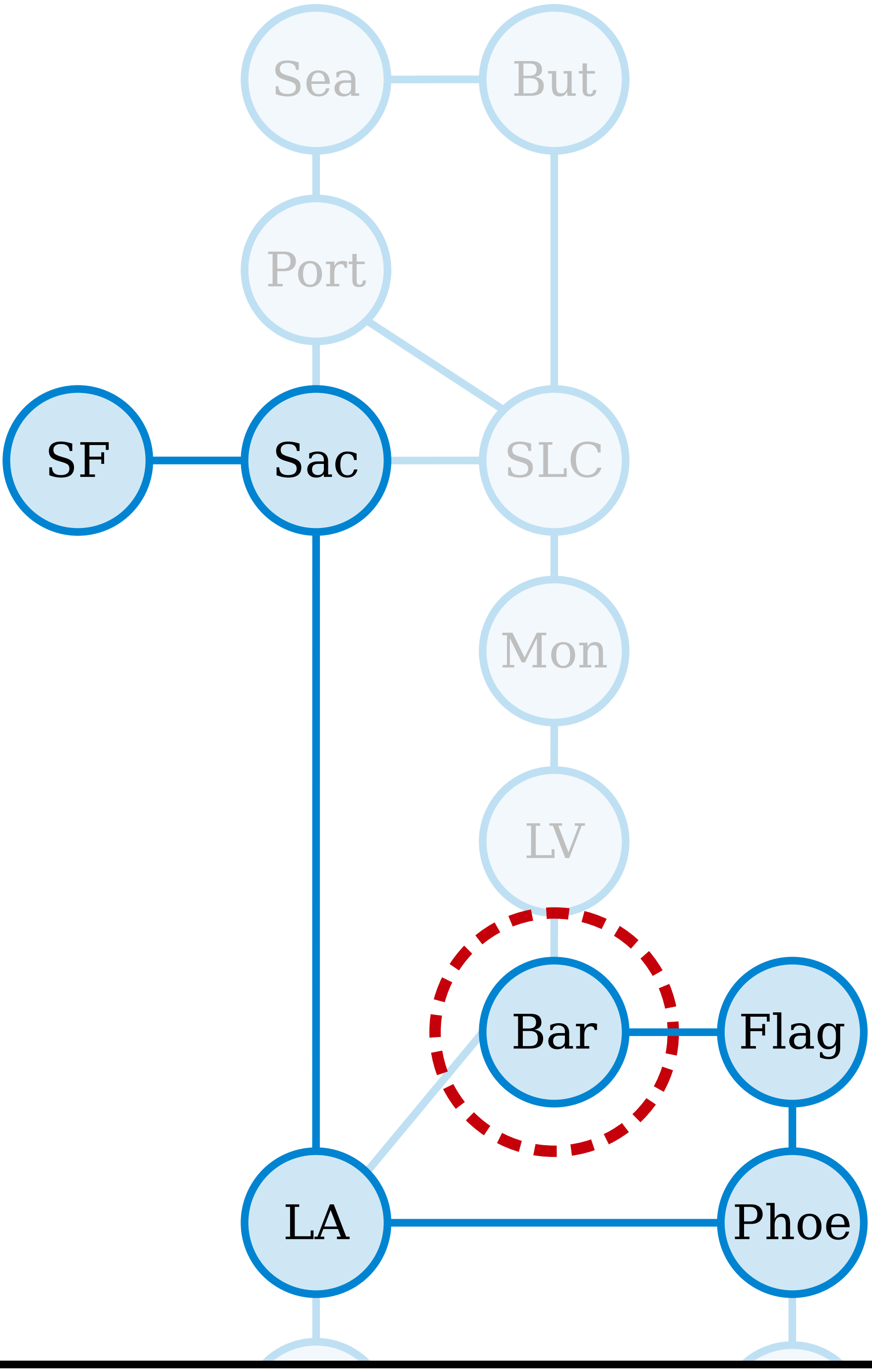


A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

SF, Sac, LA, Phoe, Flag

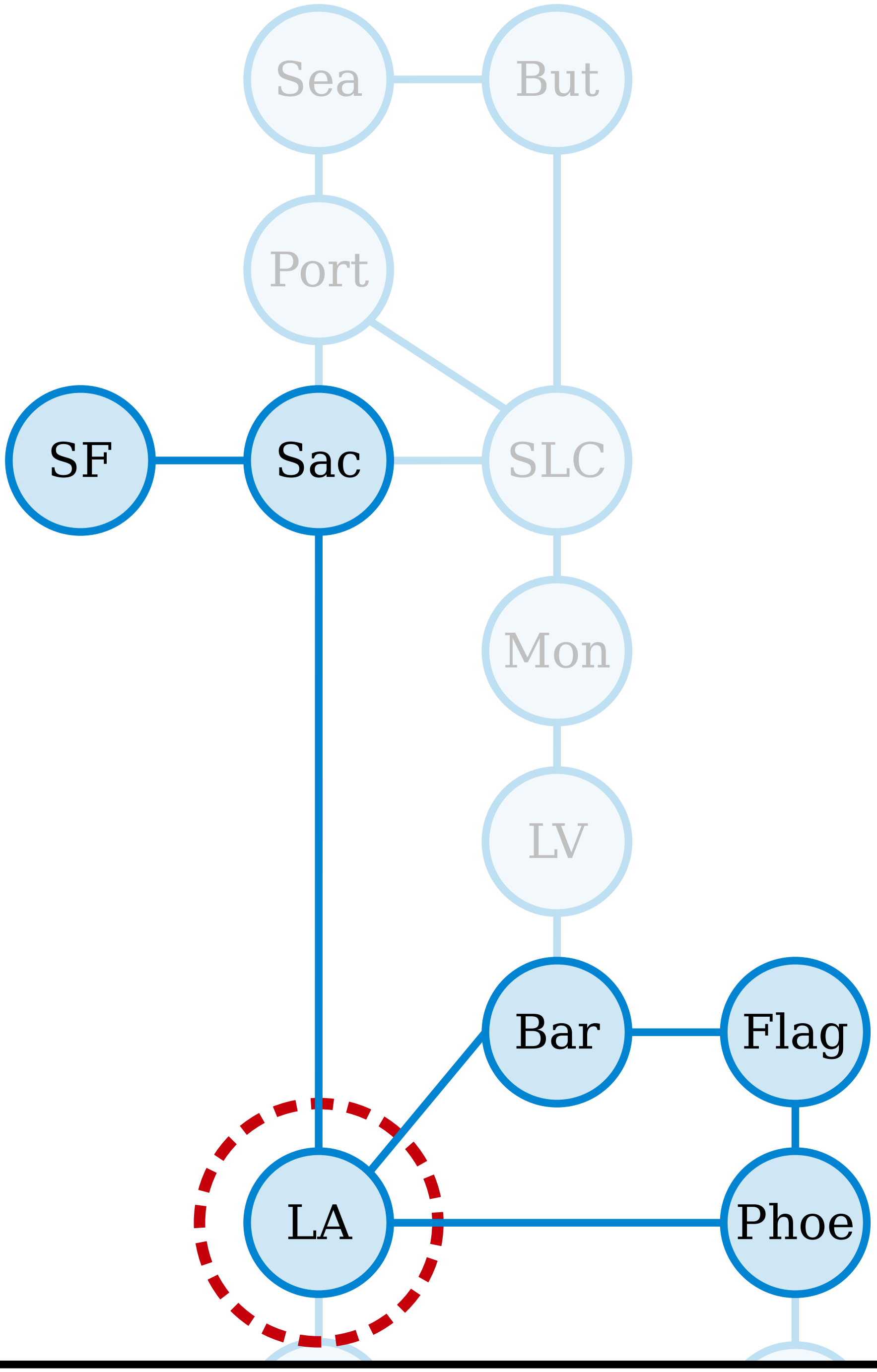


A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

SF, Sac, LA, Phoe, Flag, Bar

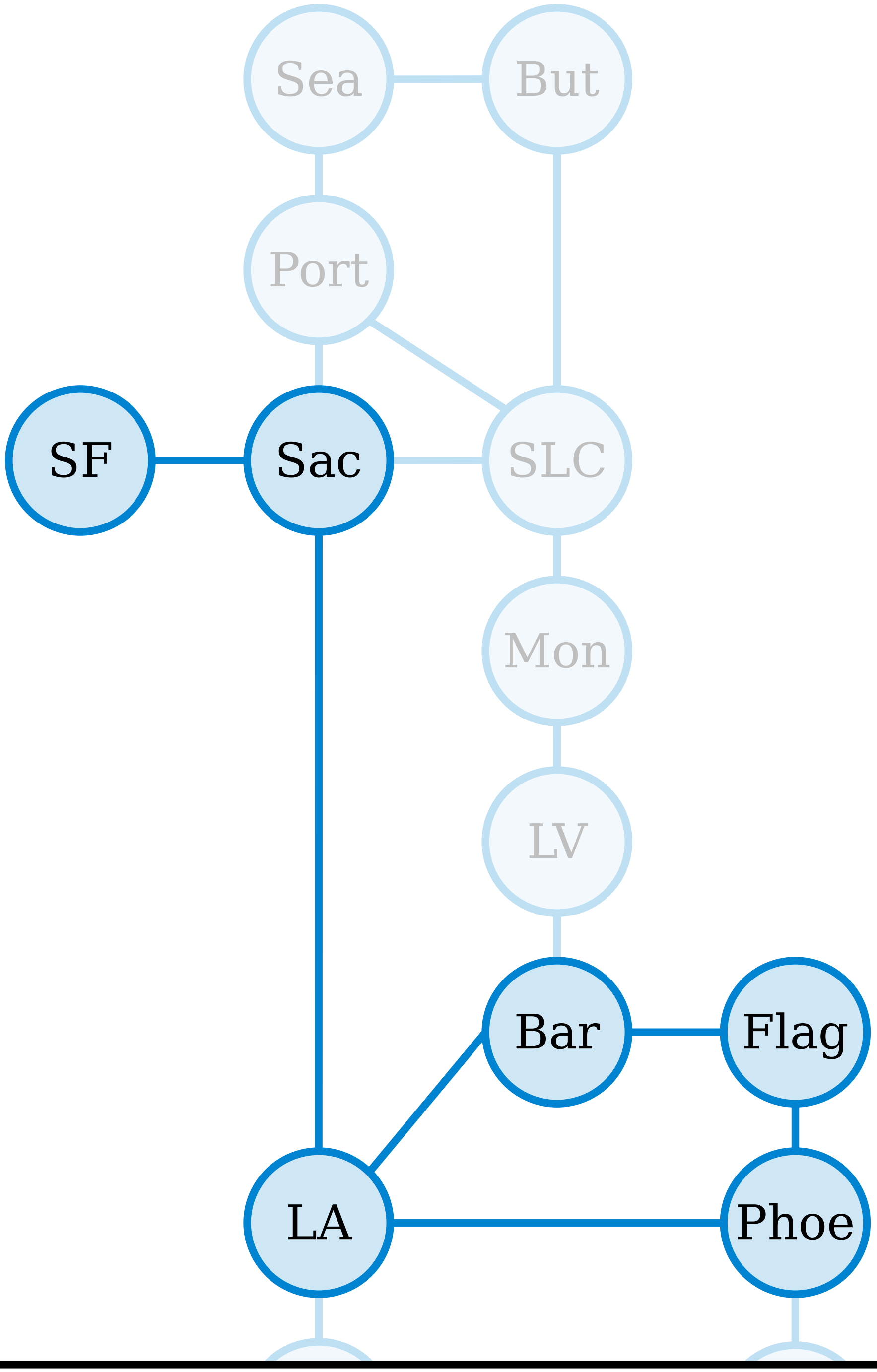


A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

SF, Sac, LA, Phoe, Flag, Bar, LA

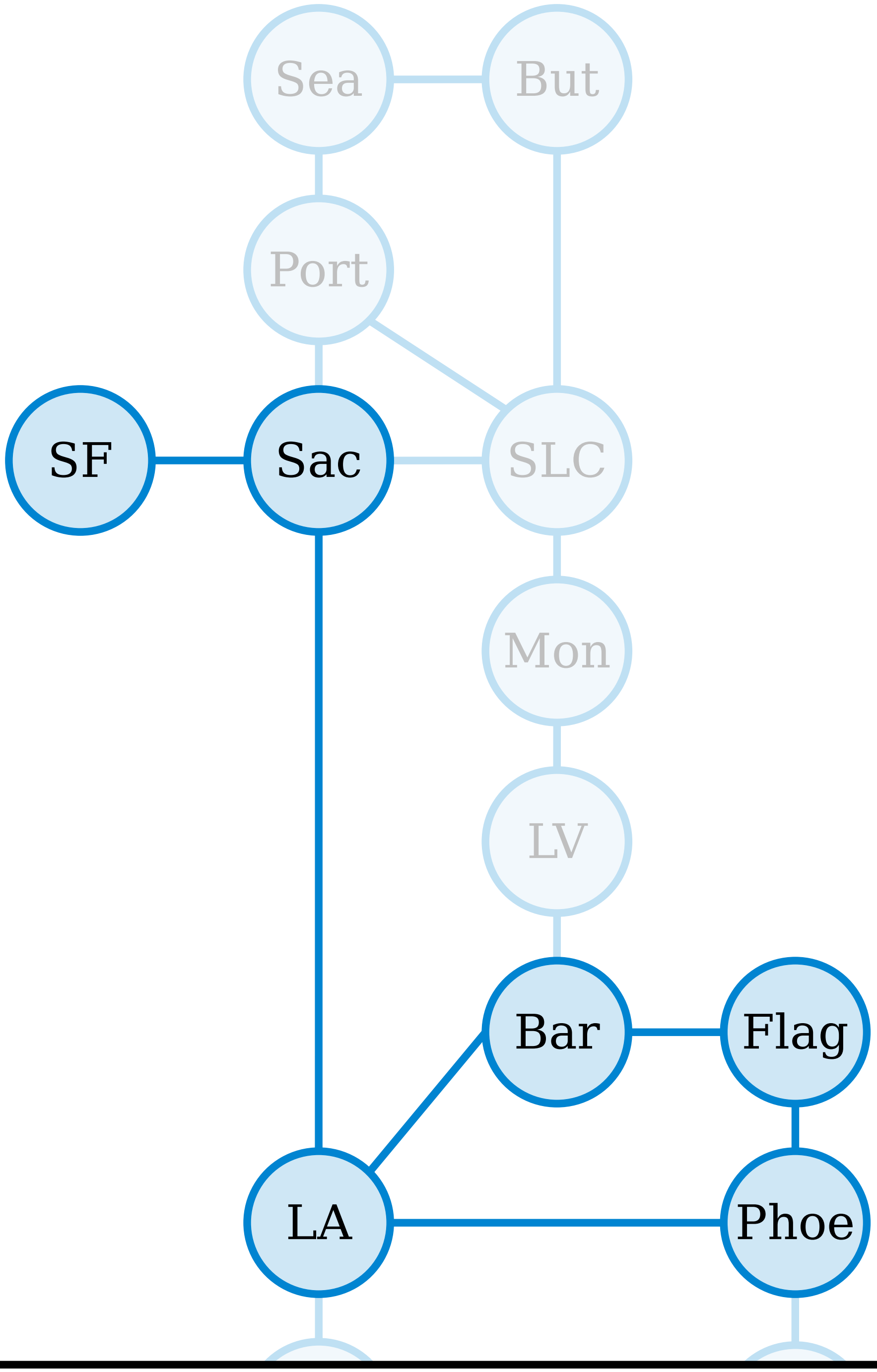


A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

SF, Sac, LA, Phoe, Flag, Bar, LA



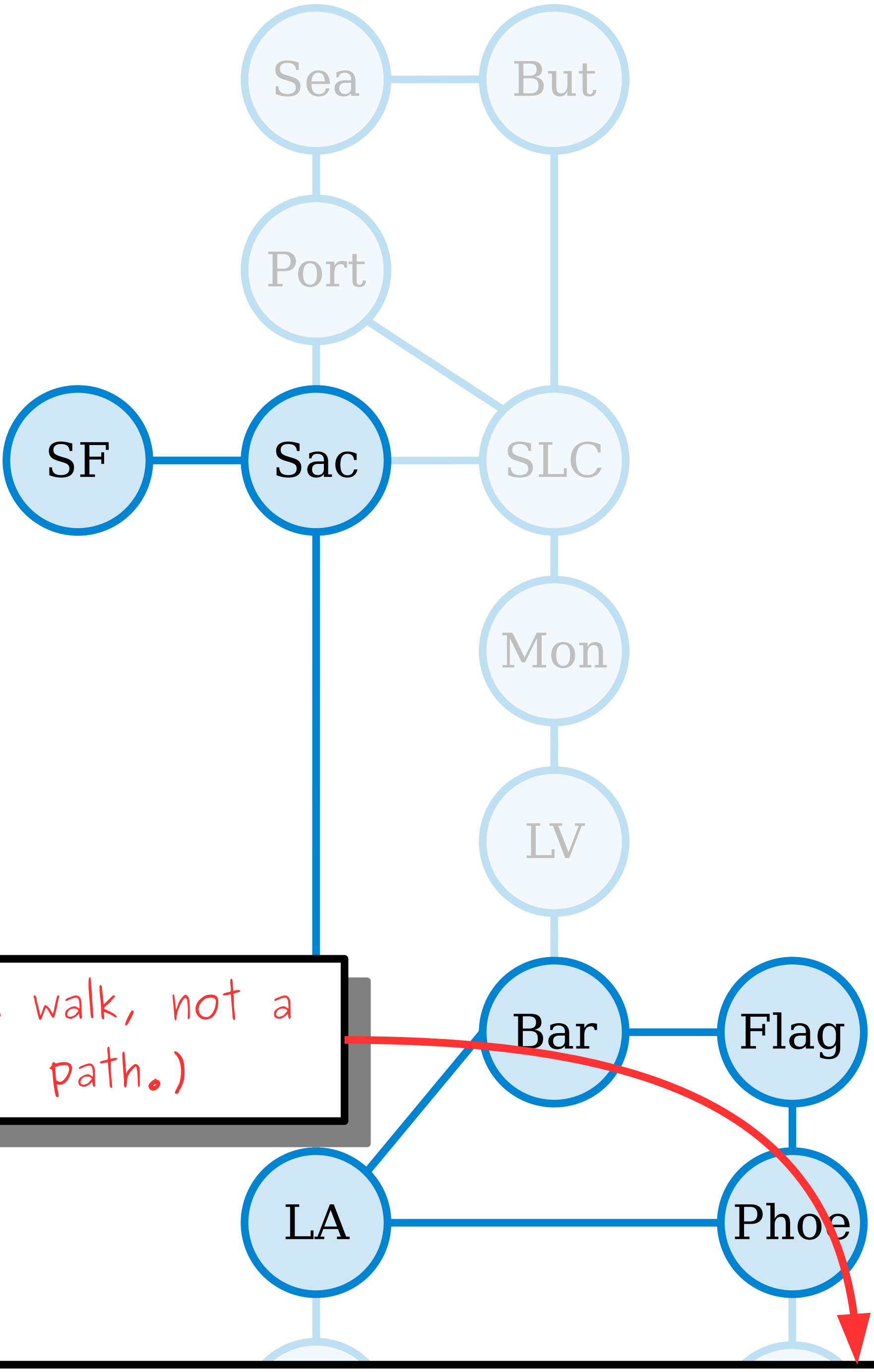
A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.

SF, Sac, LA, Phoe, Flag, Bar, LA



A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

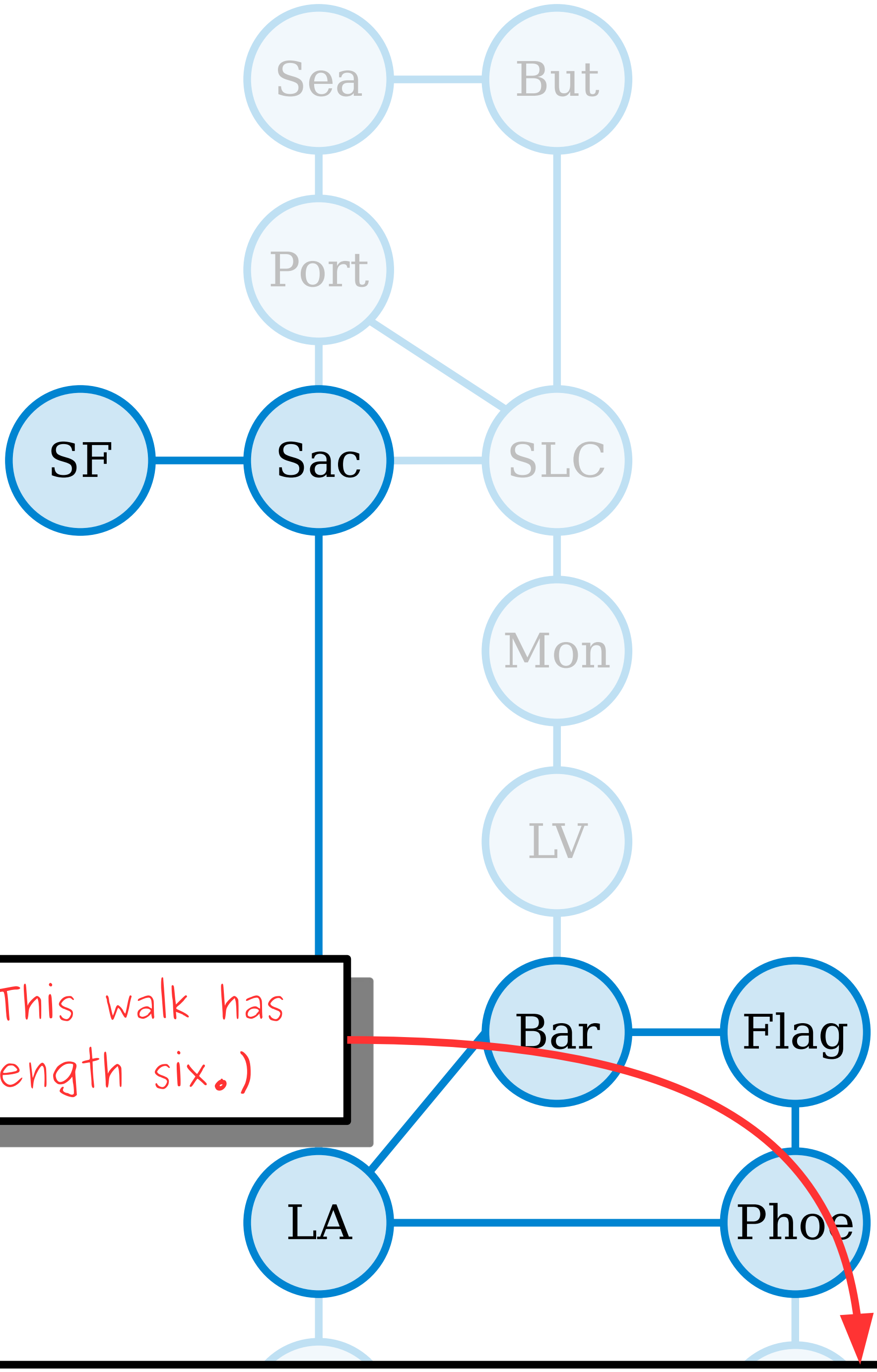
The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.

(A walk, not a path.)

SF, Sac, LA, Phoe, Flag, Bar, LA



A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

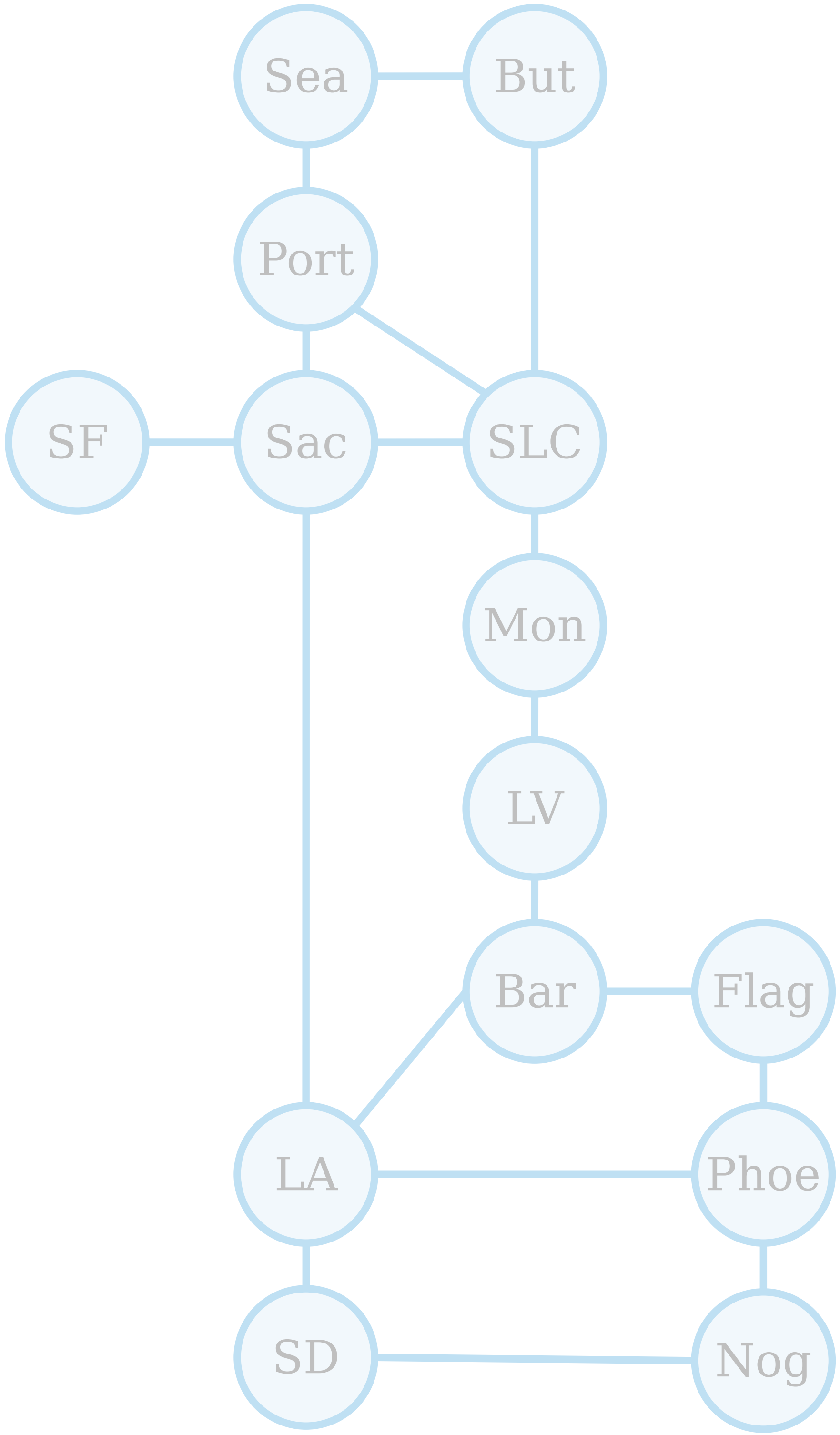
The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.

(This walk has length six.)

SF, Sac, LA, Phoe, Flag, Bar, LA

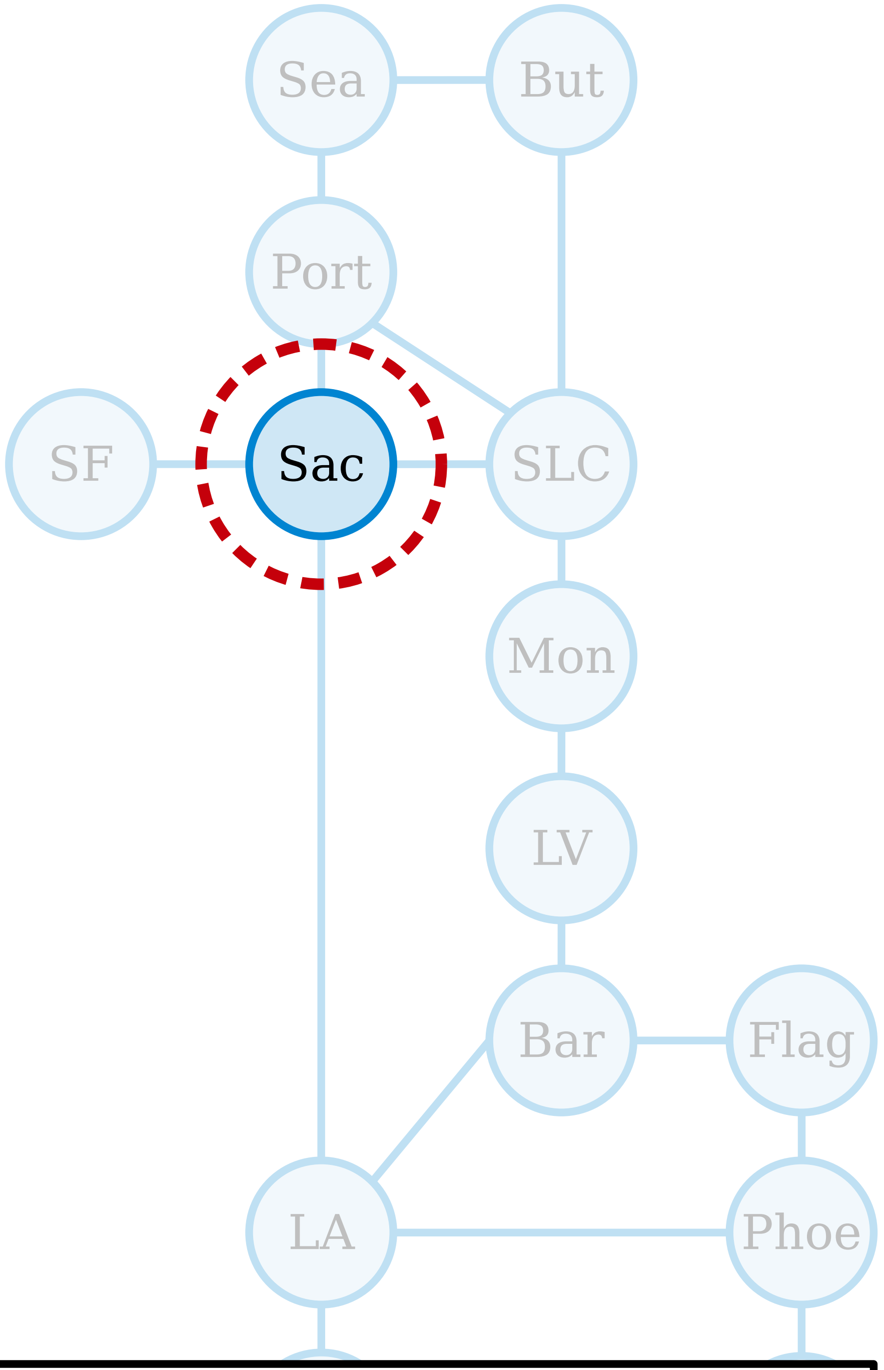


A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.



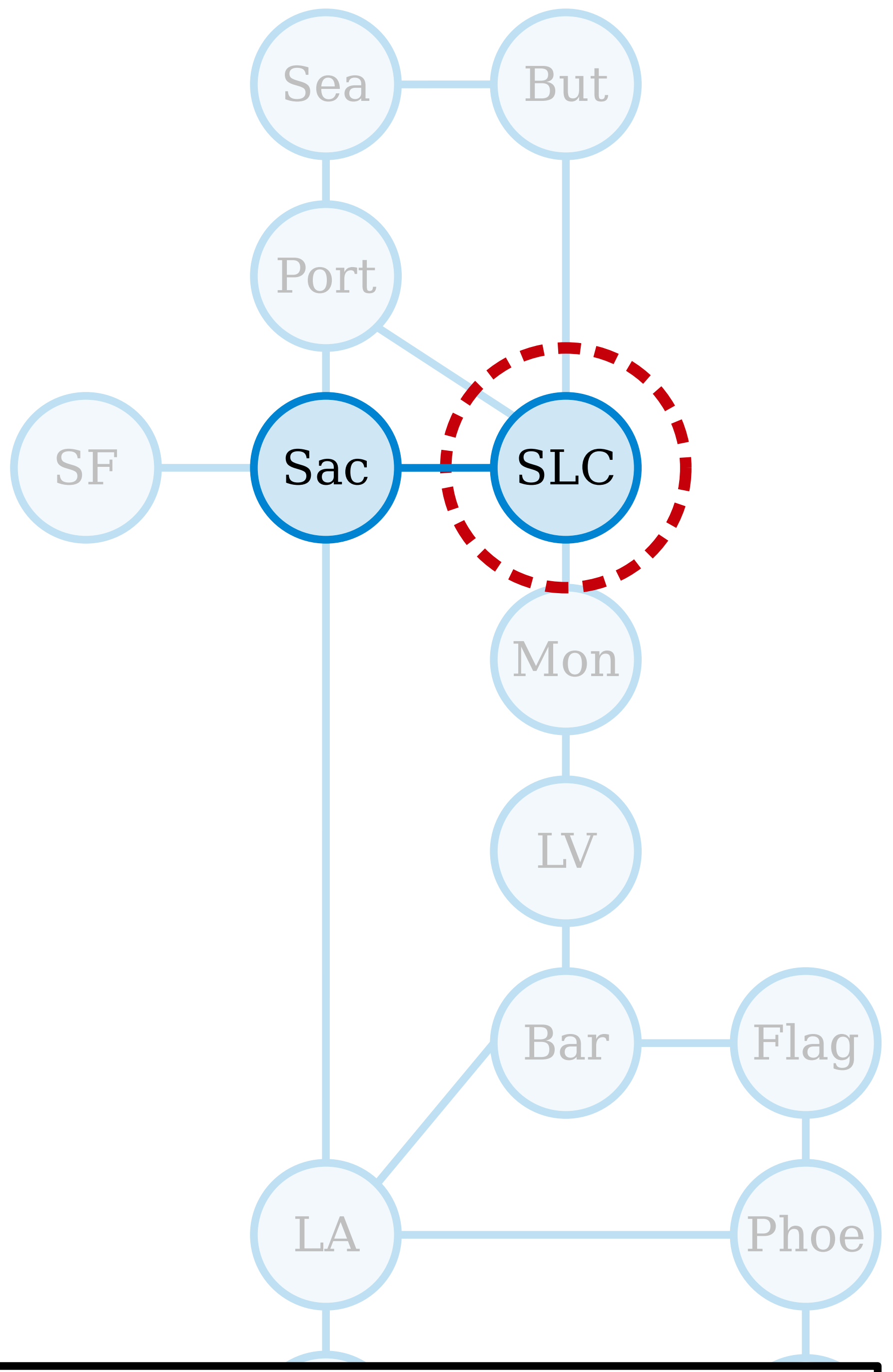
Sac

A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.



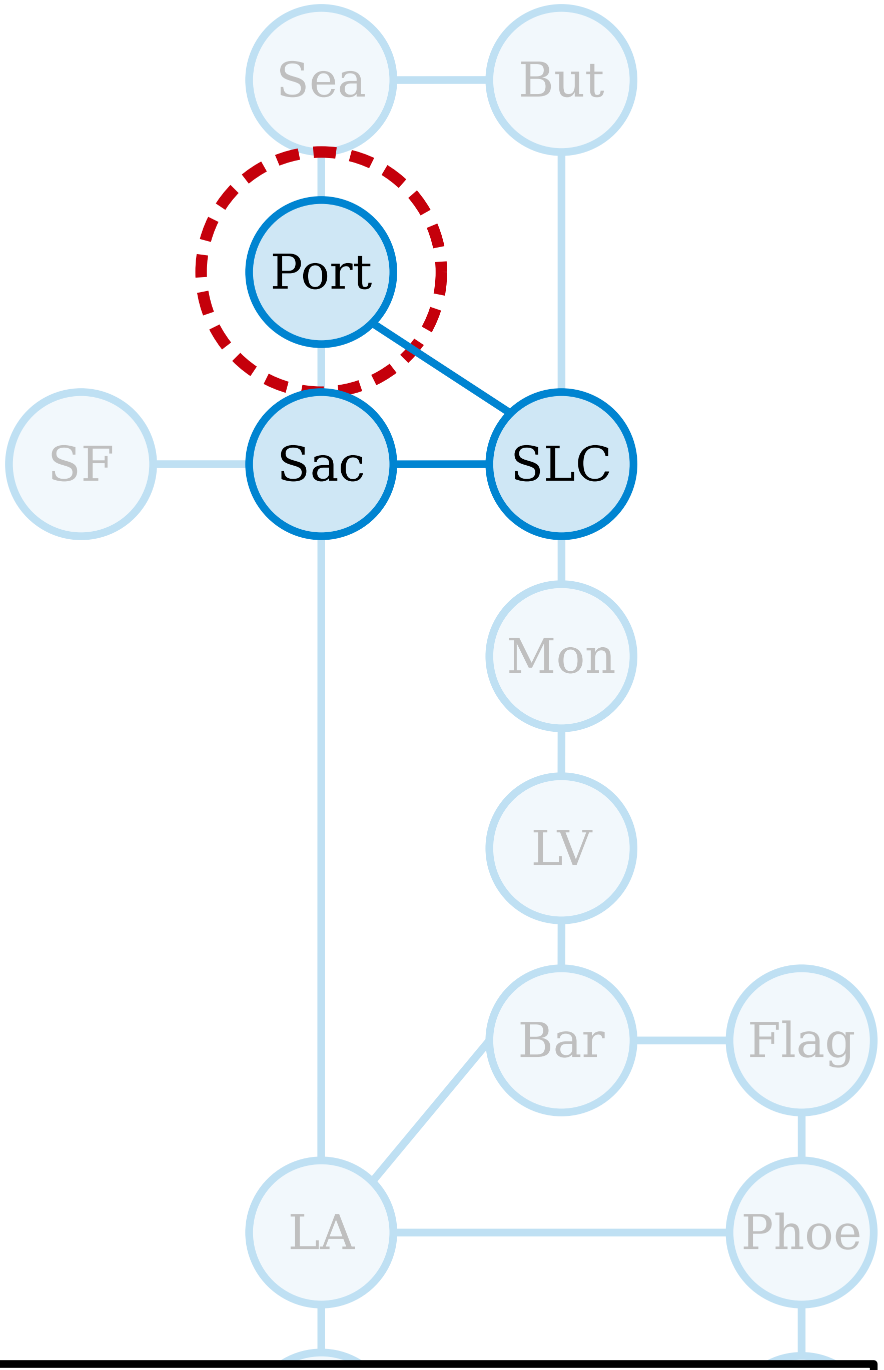
Sac, SLC

A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.



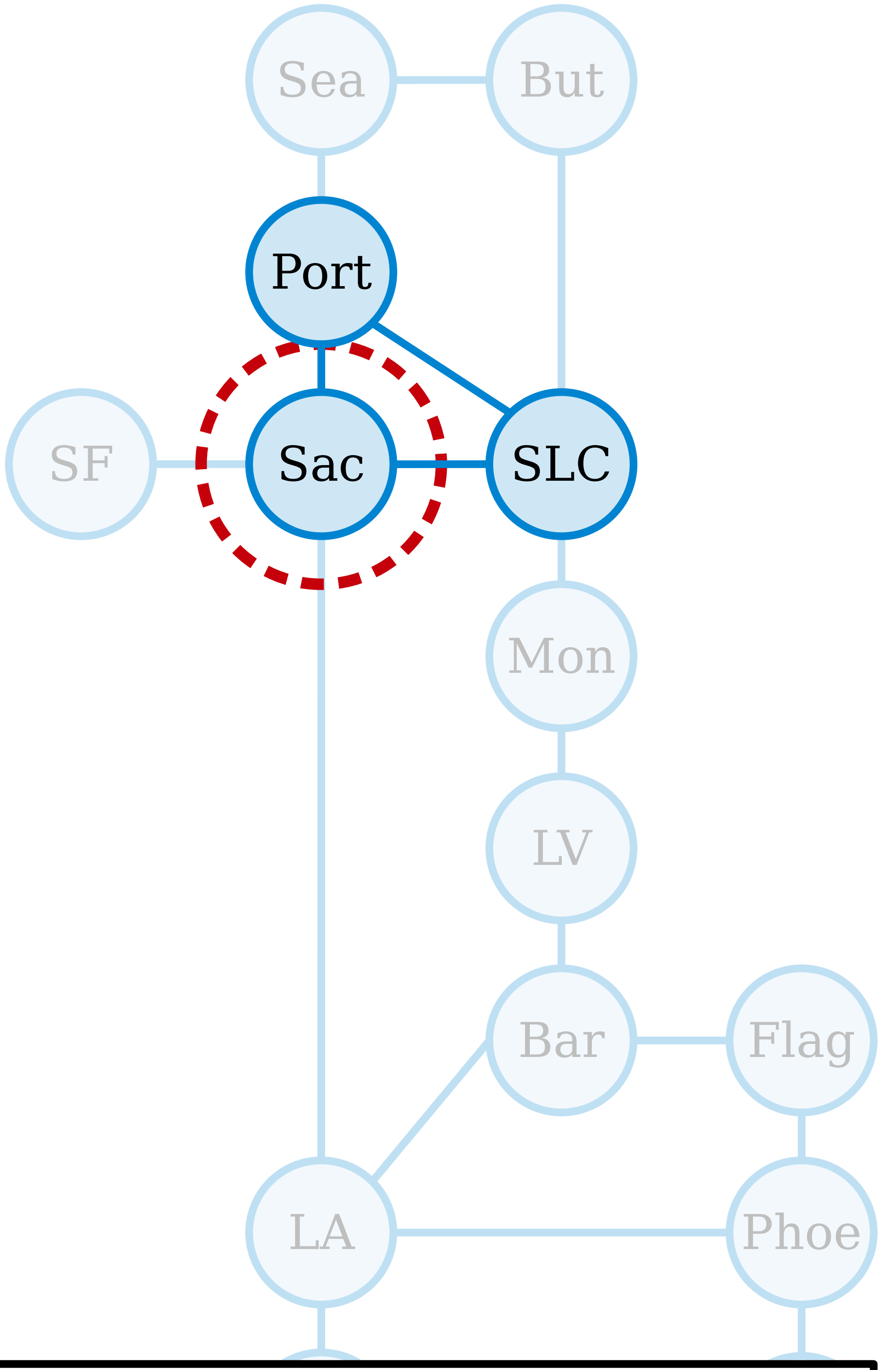
Sac, SLC, Port

A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.



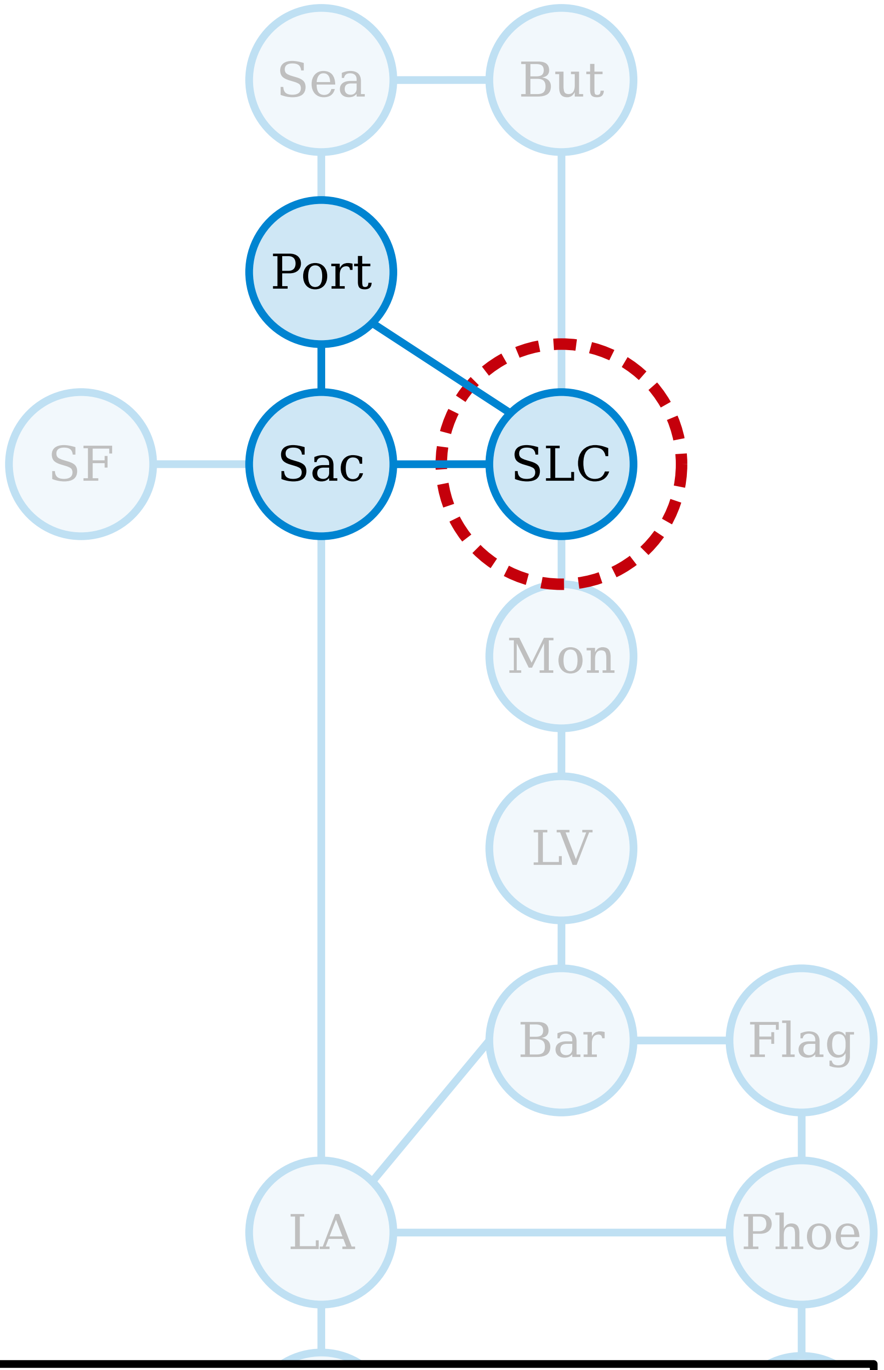
Sac, SLC, Port, Sac

A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.



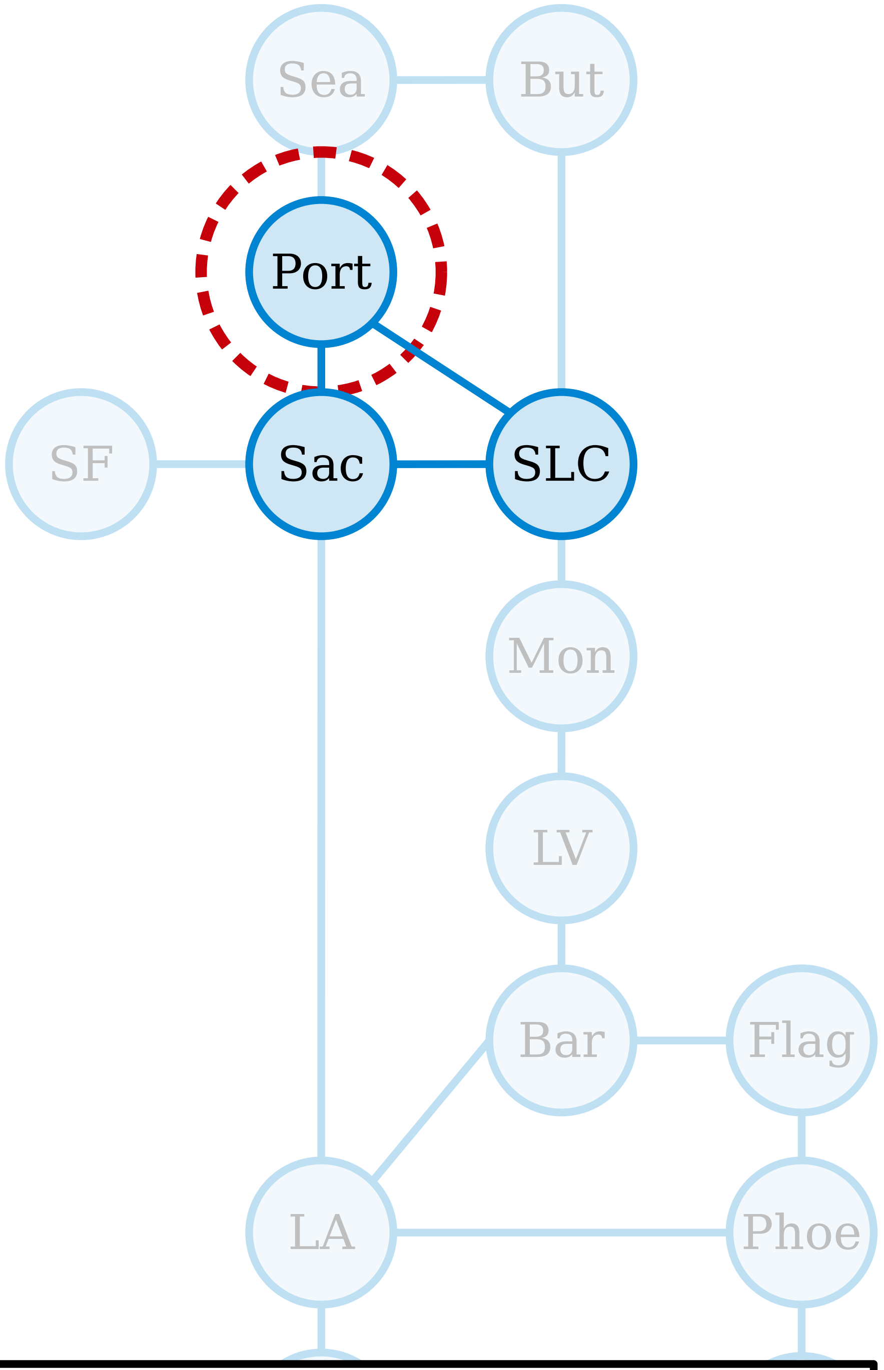
Sac, SLC, Port, Sac, SLC

A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.



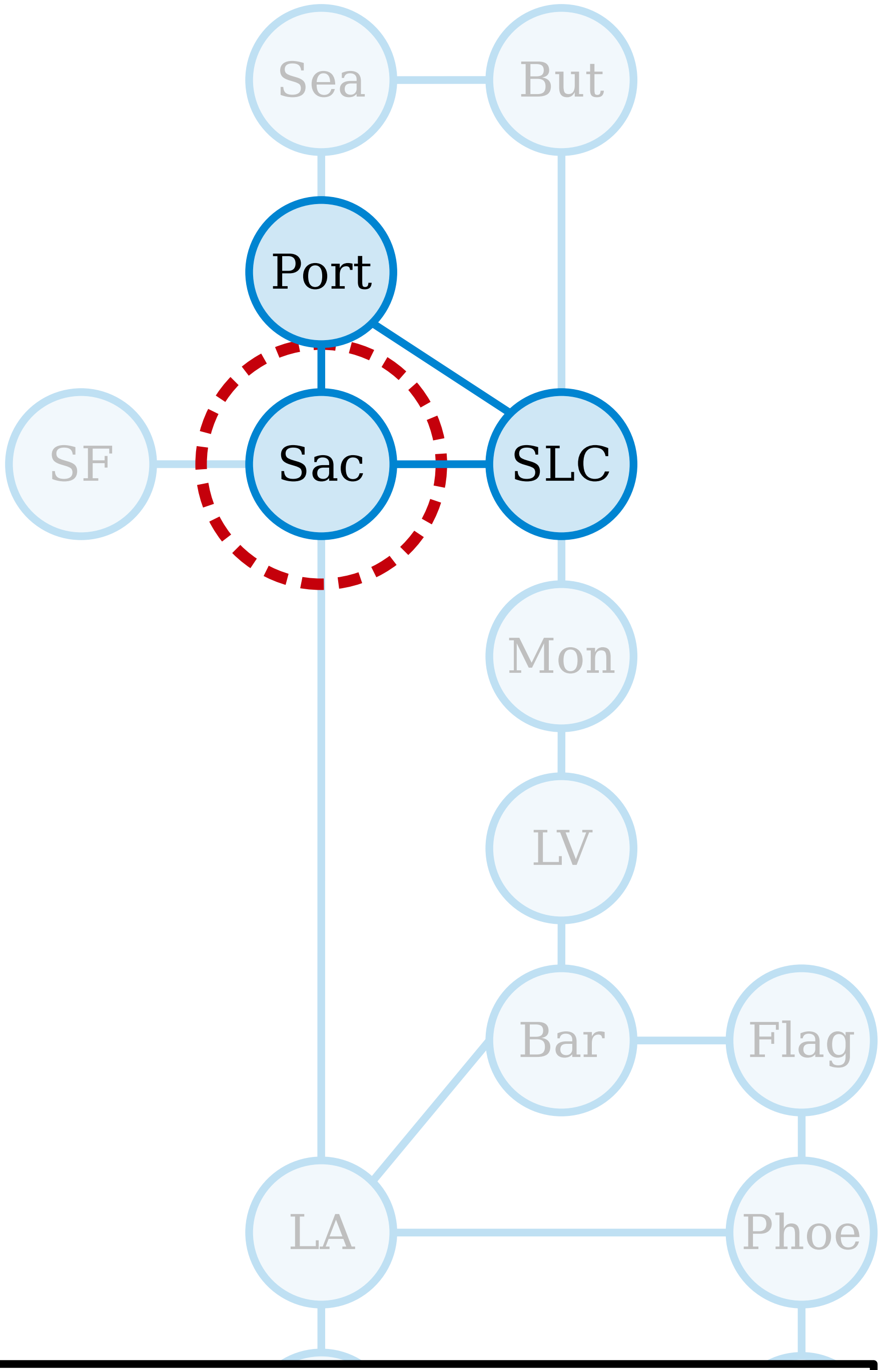
Sac, SLC, Port, Sac, SLC, Port

A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.



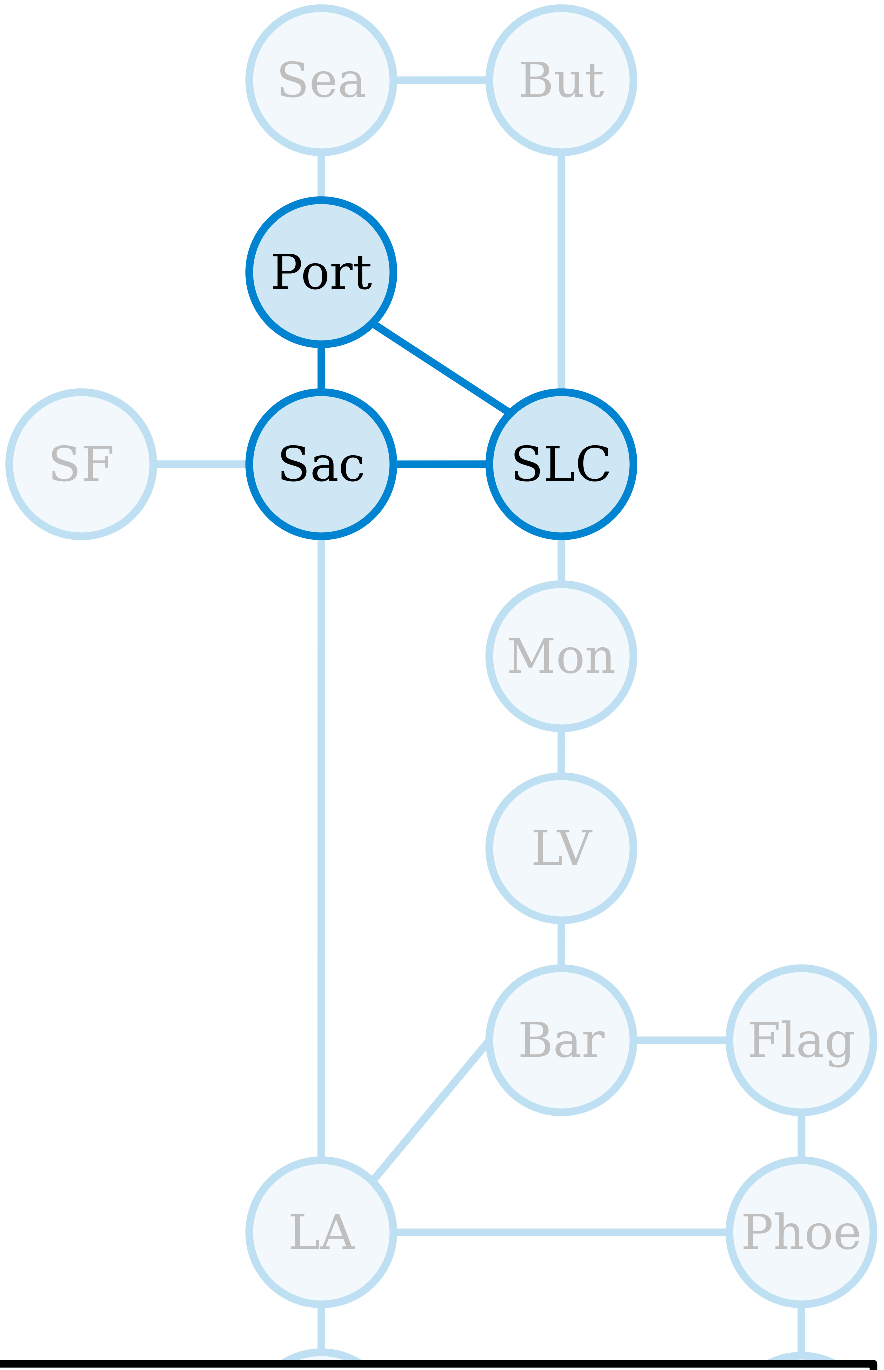
Sac, SLC, Port, Sac, SLC, Port, Sac

A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.



Sac, SLC, Port, Sac, SLC, Port, Sac

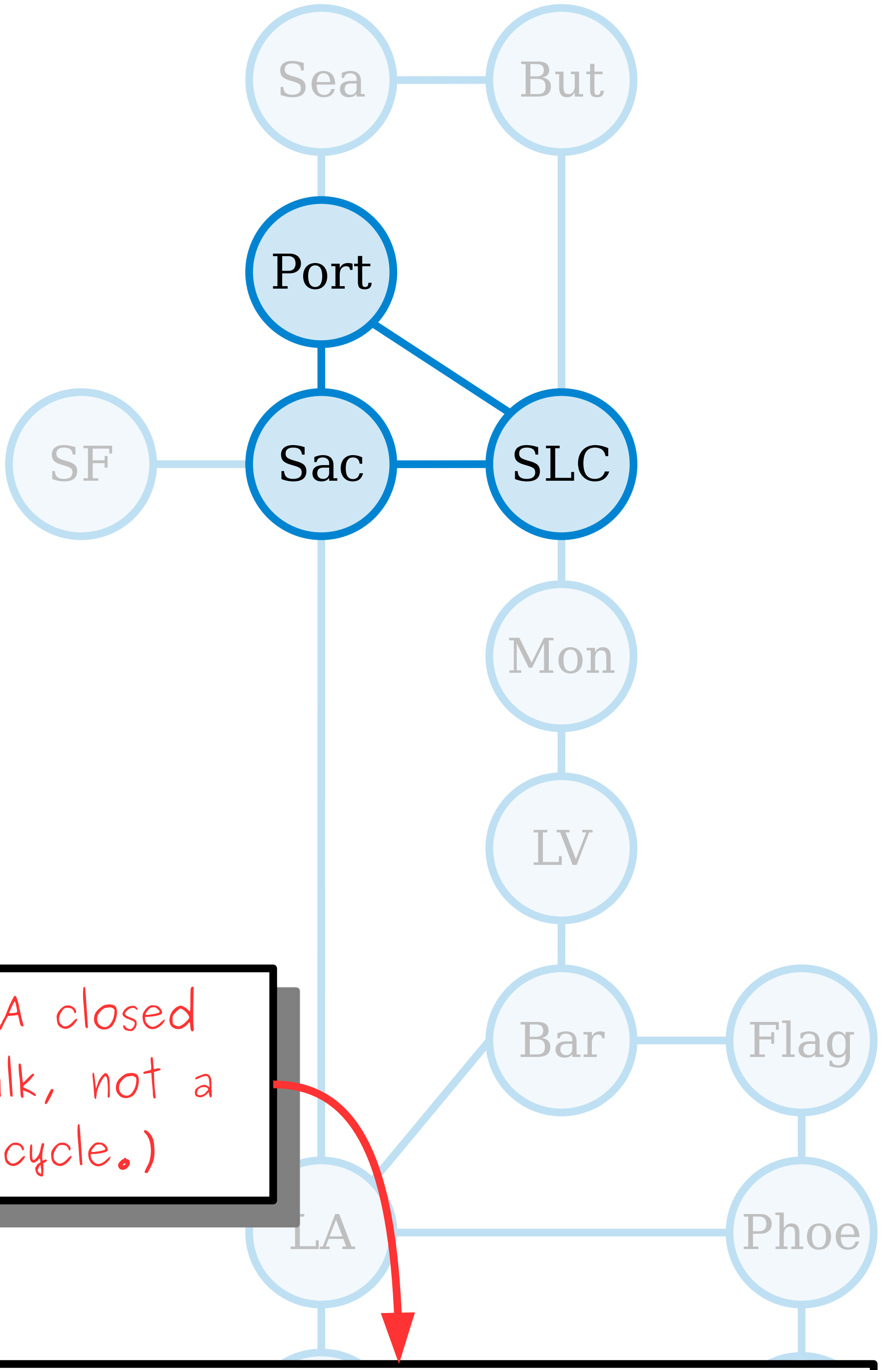
A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.

A ***cycle*** in a graph is a closed walk that does not repeat any nodes or edges except the first/last node.



(A closed walk, not a cycle.)

Sac, SLC, Port, Sac, SLC, Port, Sac

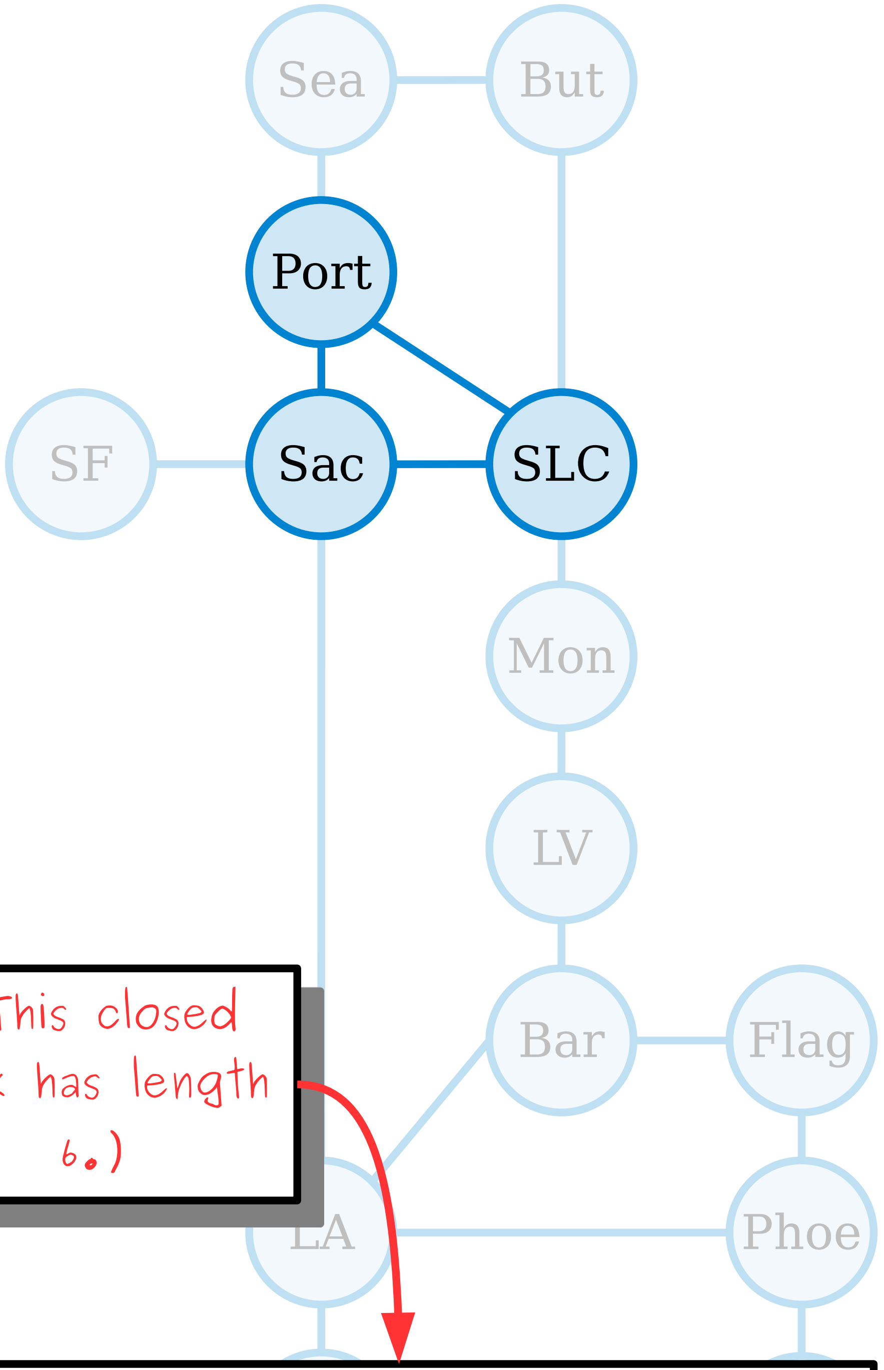
A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.

A ***cycle*** in a graph is a closed walk that does not repeat any nodes or edges except the first/last node.



(This closed walk has length 6.)

Sac, SLC, Port, Sac, SLC, Port, Sac

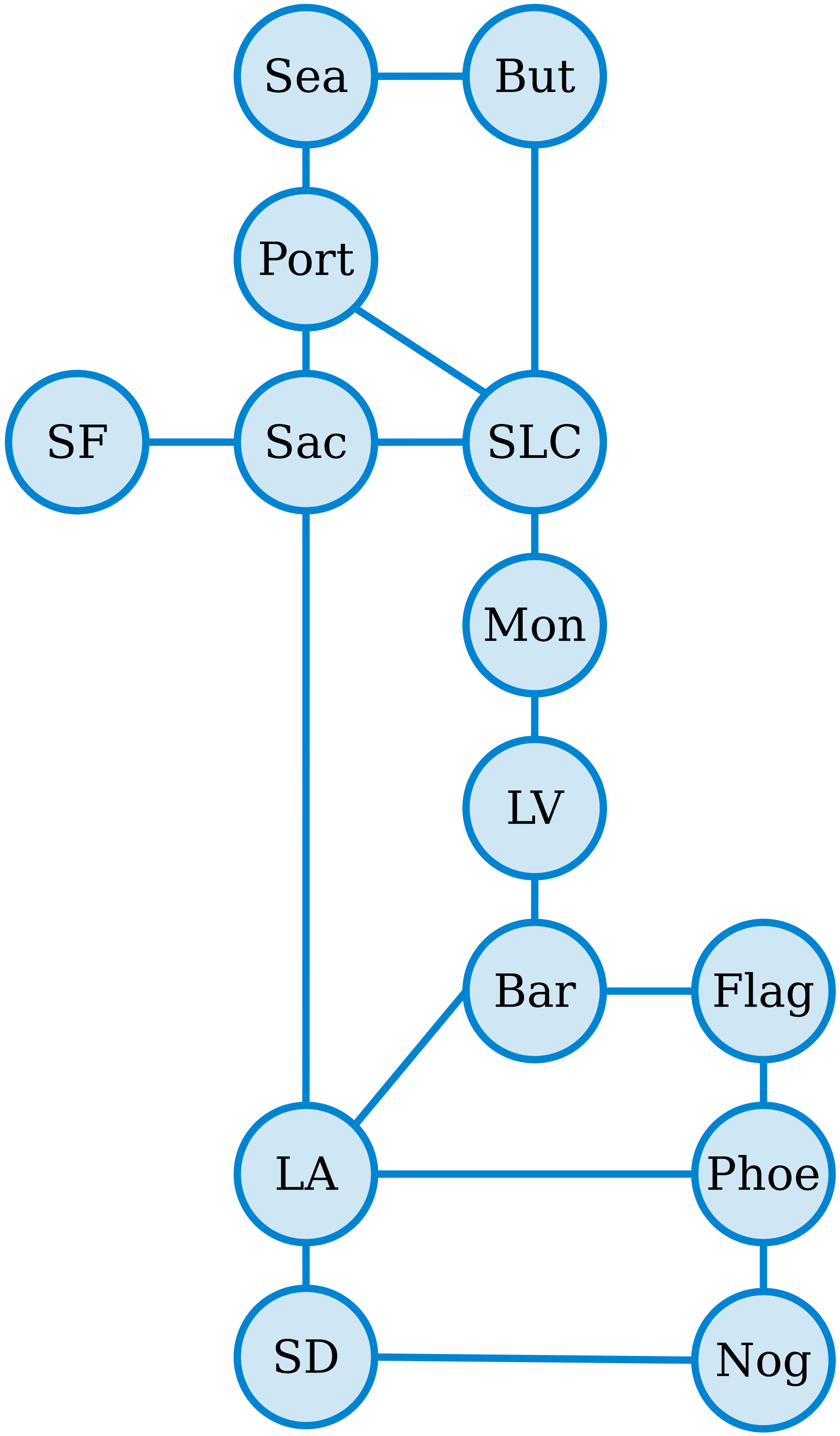
A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.

A ***cycle*** in a graph is a closed walk that does not repeat any nodes or edges except the first/last node.



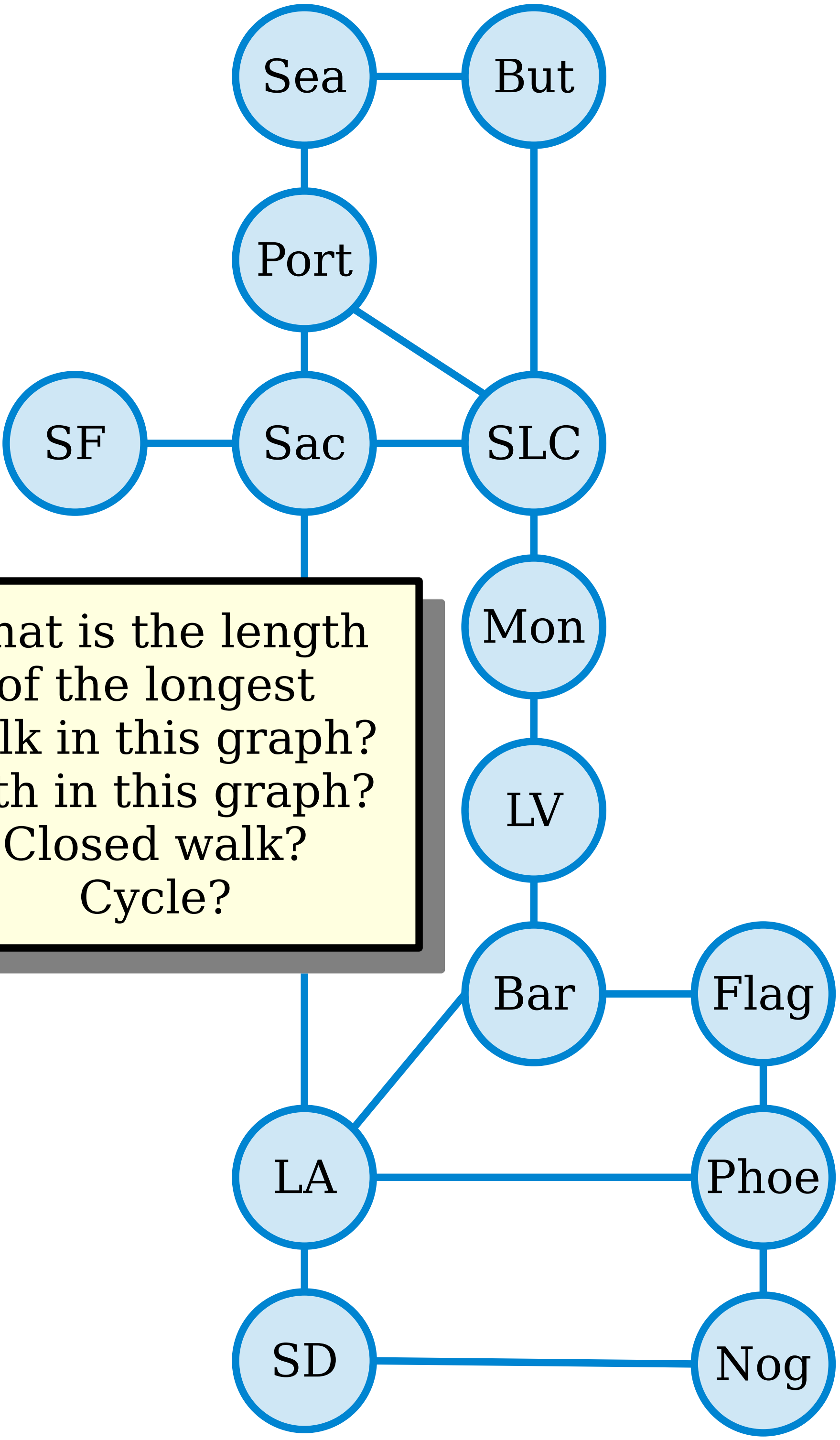
A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

A ***path*** in a graph is walk that does not repeat any nodes.

A ***cycle*** in a graph is a closed walk that does not repeat any nodes or edges except the first/last node.



What is the length of the longest walk in this graph? Path in this graph? Closed walk? Cycle?

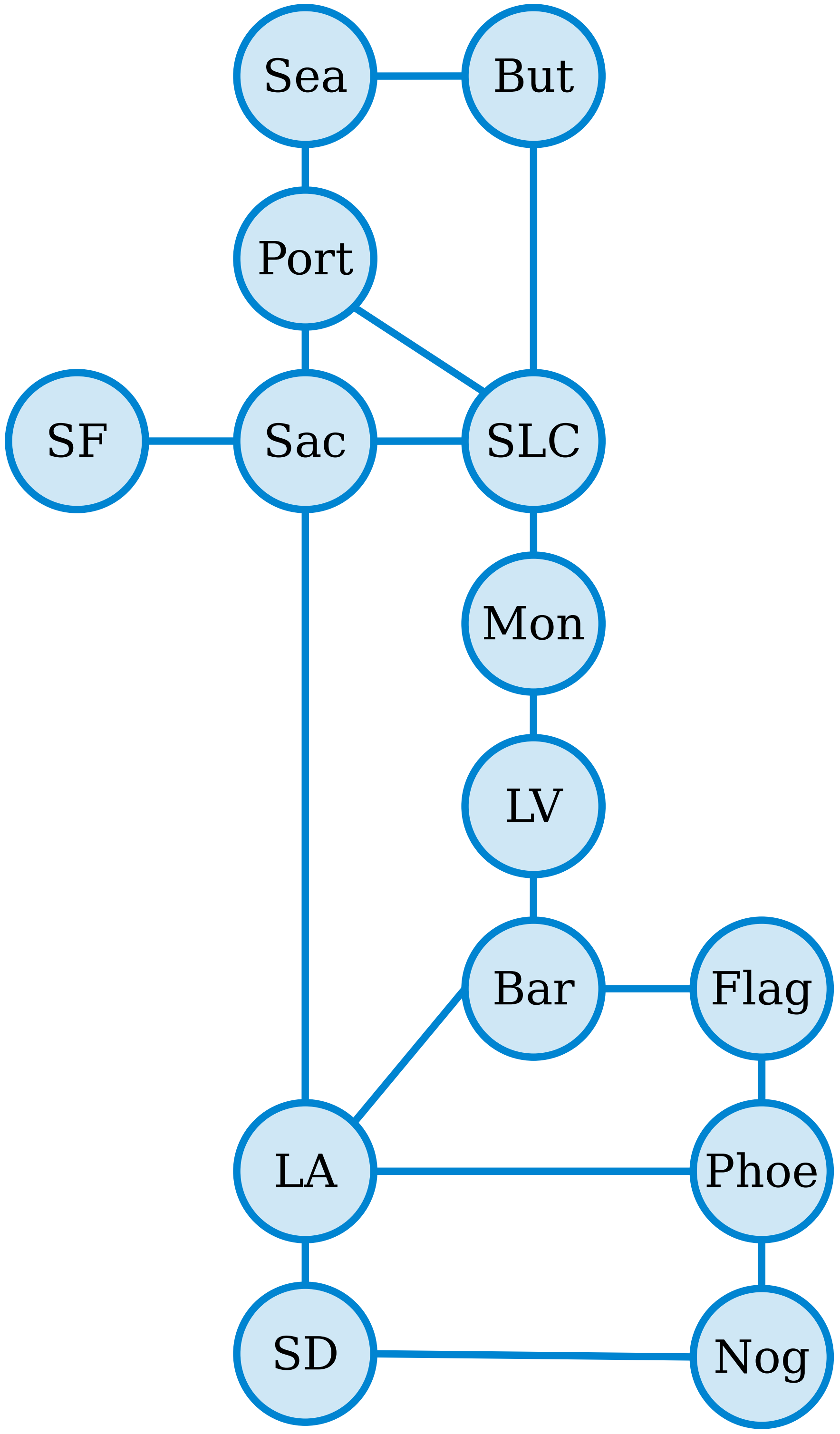
A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the walk $v_1, \ldots, v_n$ is $n - 1$.

A ***closed walk*** in a graph is a walk from a node back to itself. (By convention, a closed walk cannot have length zero.)

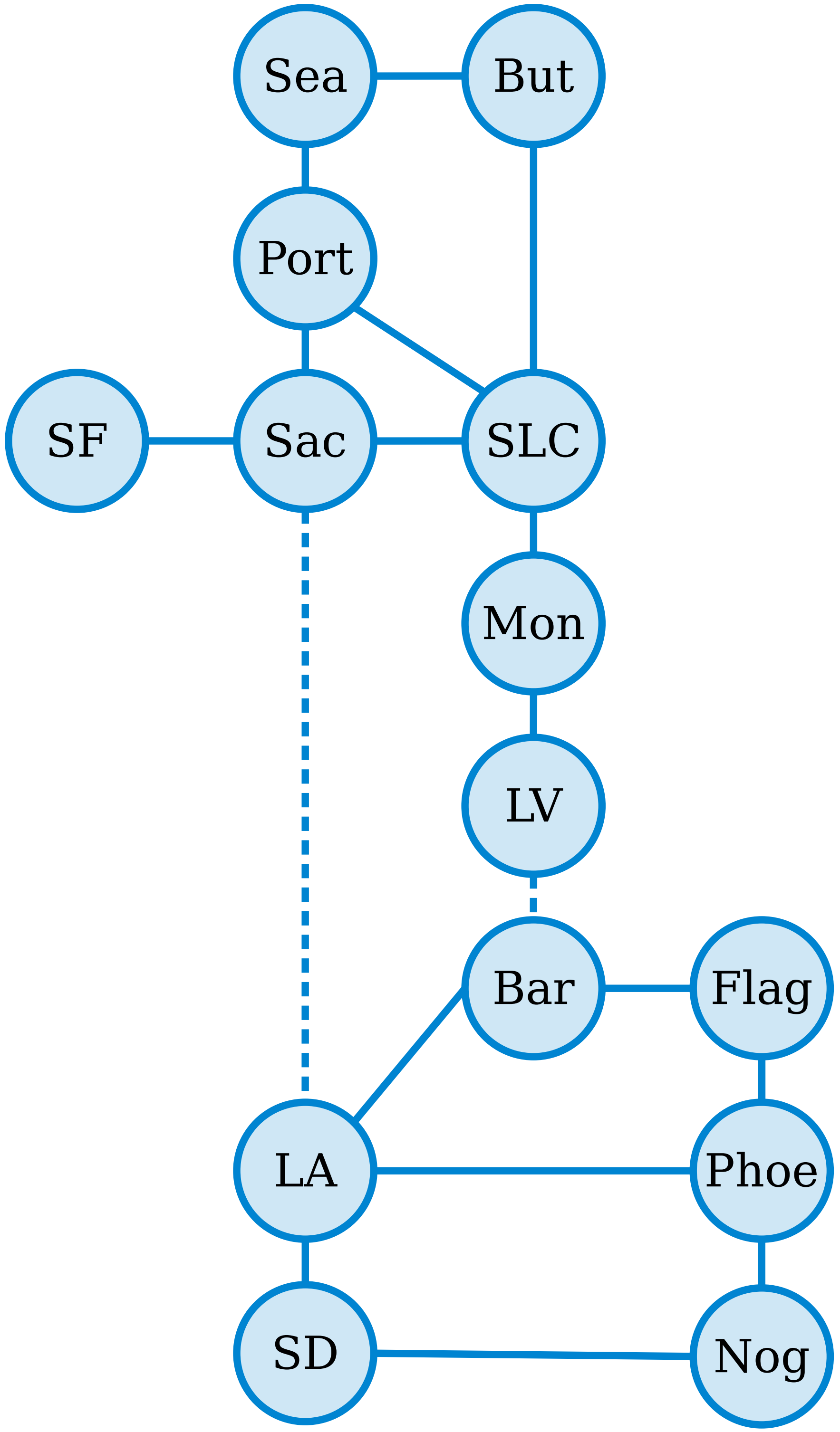
A ***path*** in a graph is walk that does not repeat any nodes.

A ***cycle*** in a graph is a closed walk that does not repeat any nodes or edges except the first/last node.



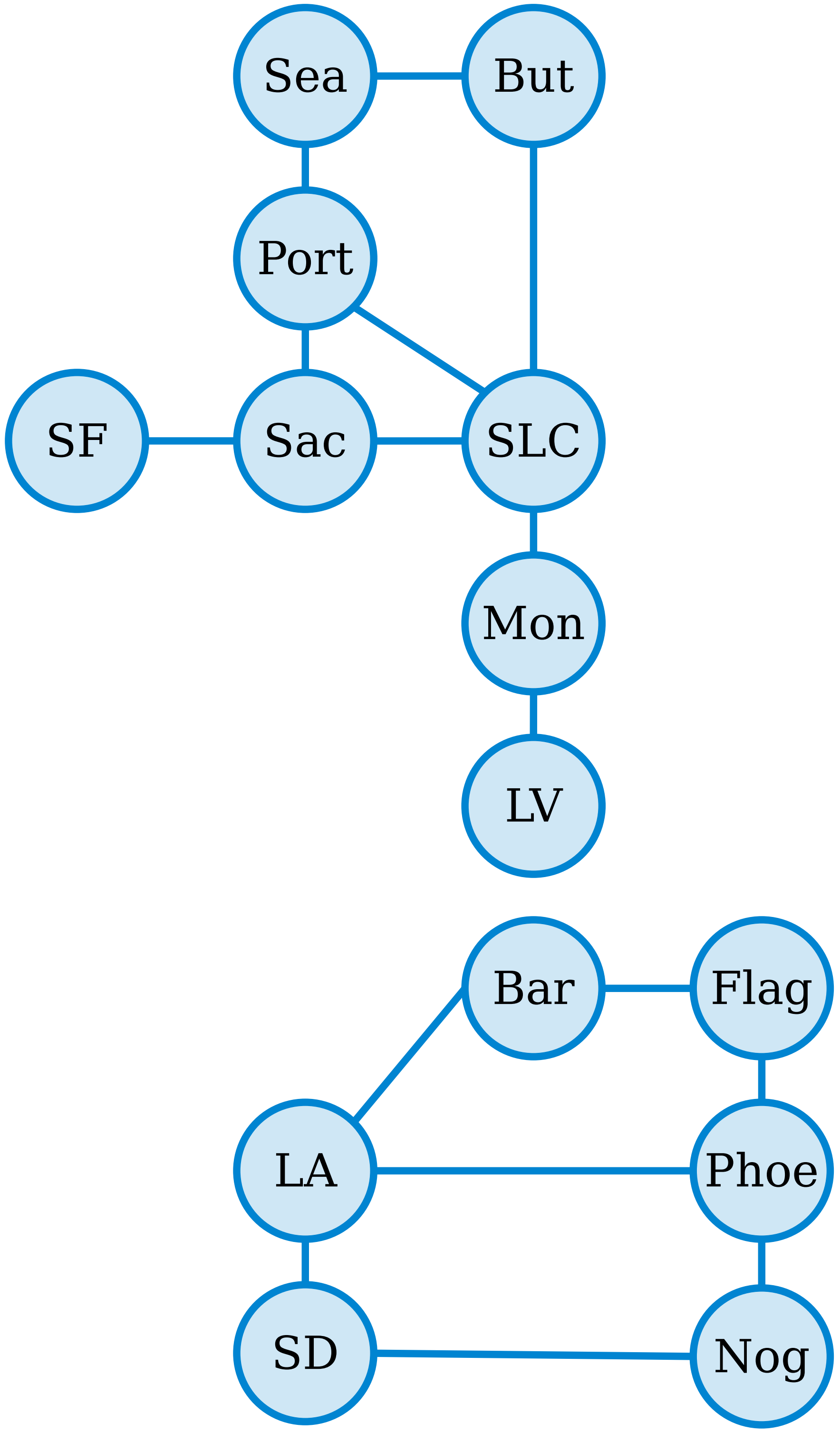
A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

A ***path*** in a graph is walk that does not repeat any nodes.



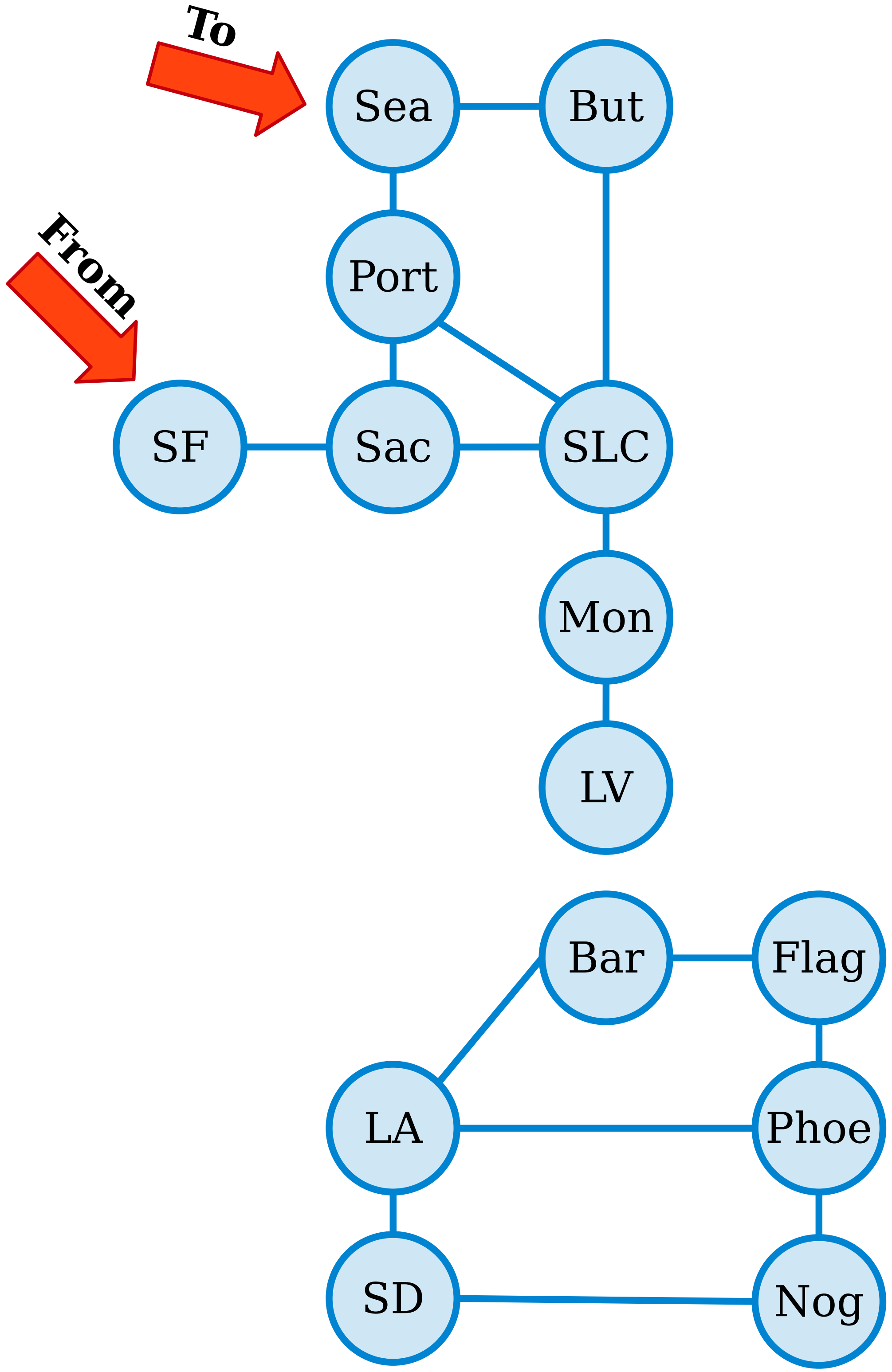
A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

A ***path*** in a graph is walk that does not repeat any nodes.



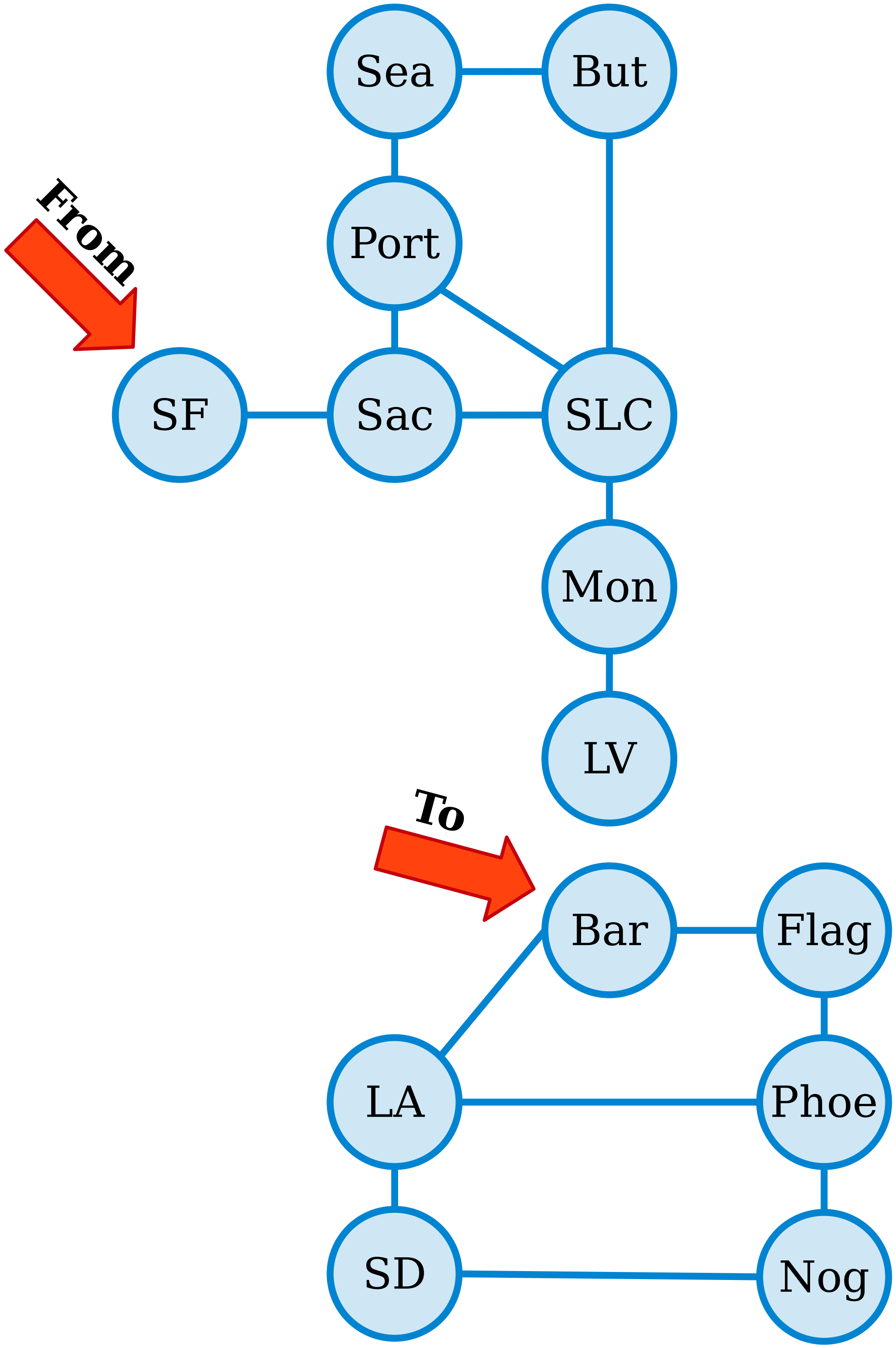
A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

A ***path*** in a graph is walk that does not repeat any nodes.



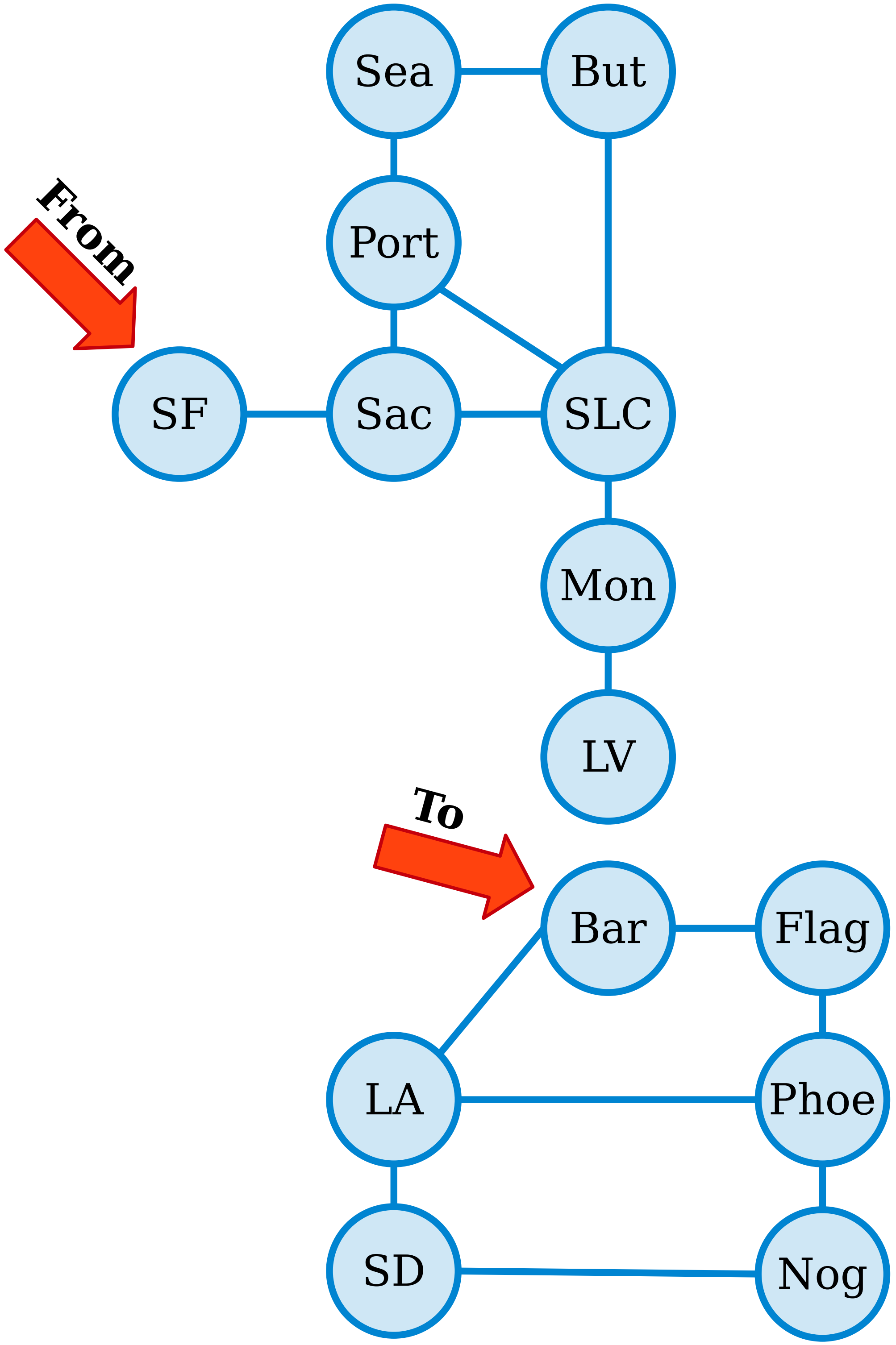
A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

A ***path*** in a graph is walk that does not repeat any nodes.



A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

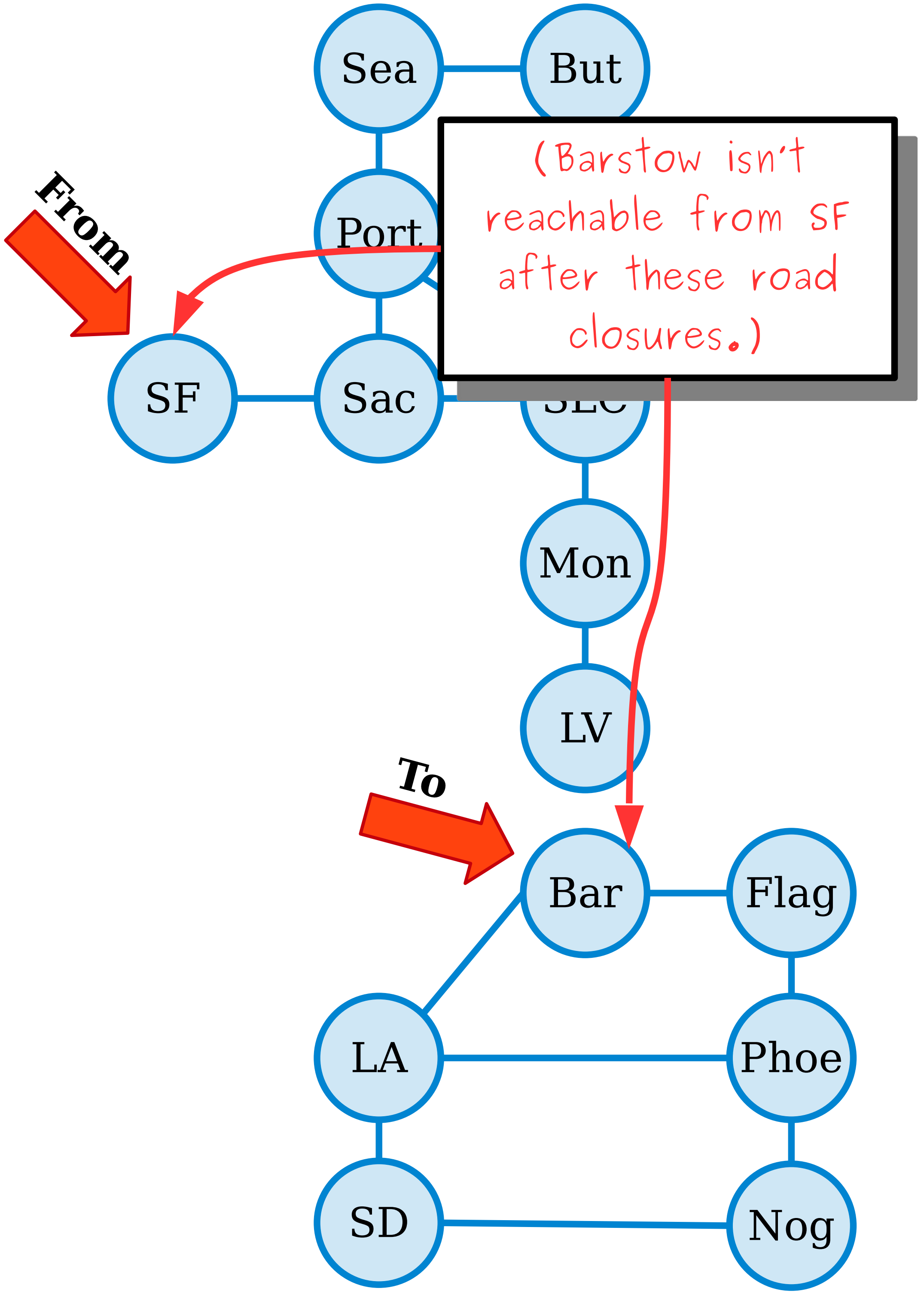
A ***path*** in a graph is walk that does not repeat any nodes.



A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

A ***path*** in a graph is walk that does not repeat any nodes.

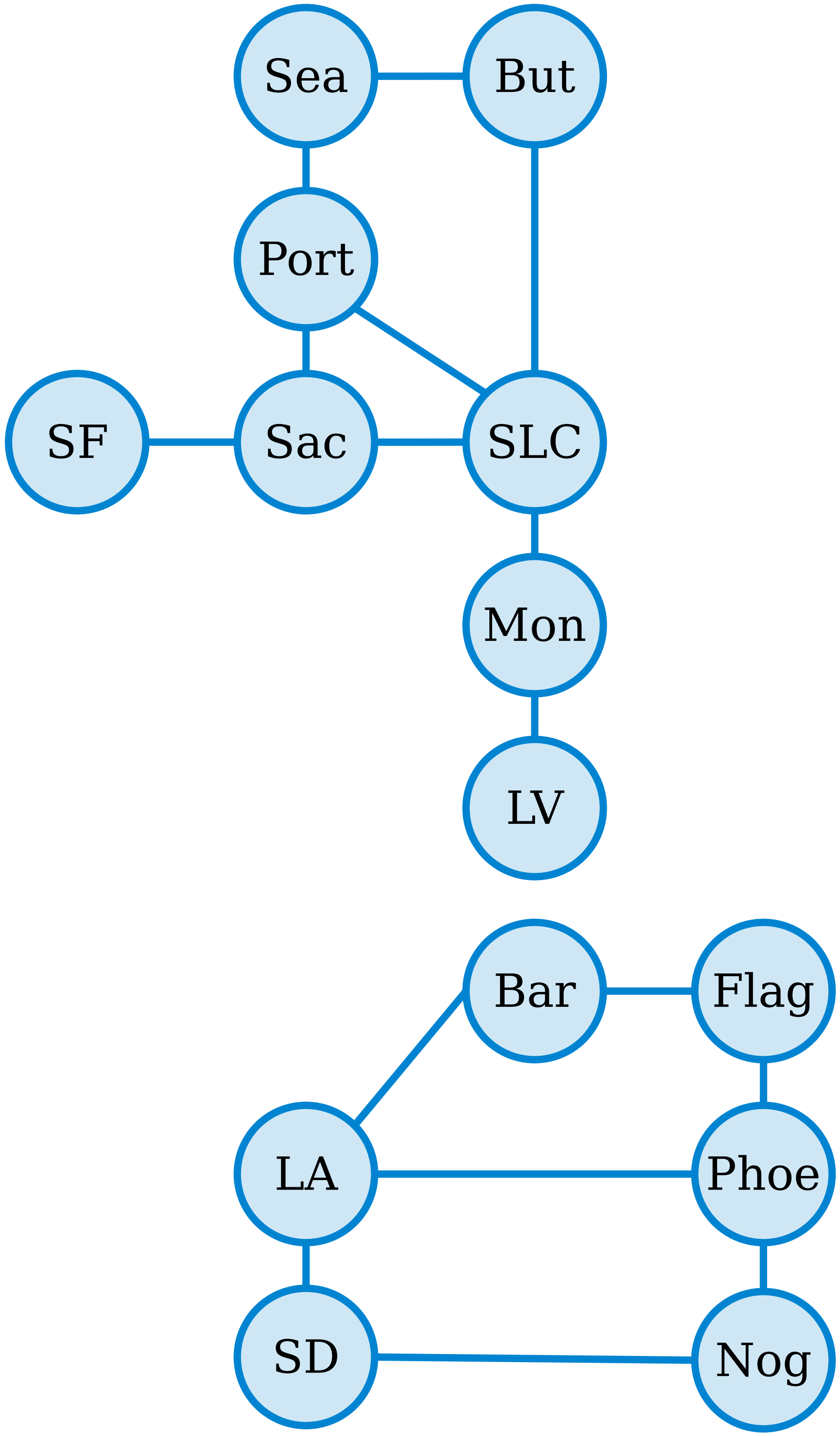
A node v is ***reachable*** from a node u if there is a path from u to v.


Sea
But
Port
SF
Sac
SLC
Mon
LV
Bar
Flag
LA
Phoe
SD
Nog
From
To
(Barstow isn't reachable from SF after these road closures.)

A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

A ***path*** in a graph is walk that does not repeat any nodes.

A node v is ***reachable*** from a node u if there is a path from u to v.

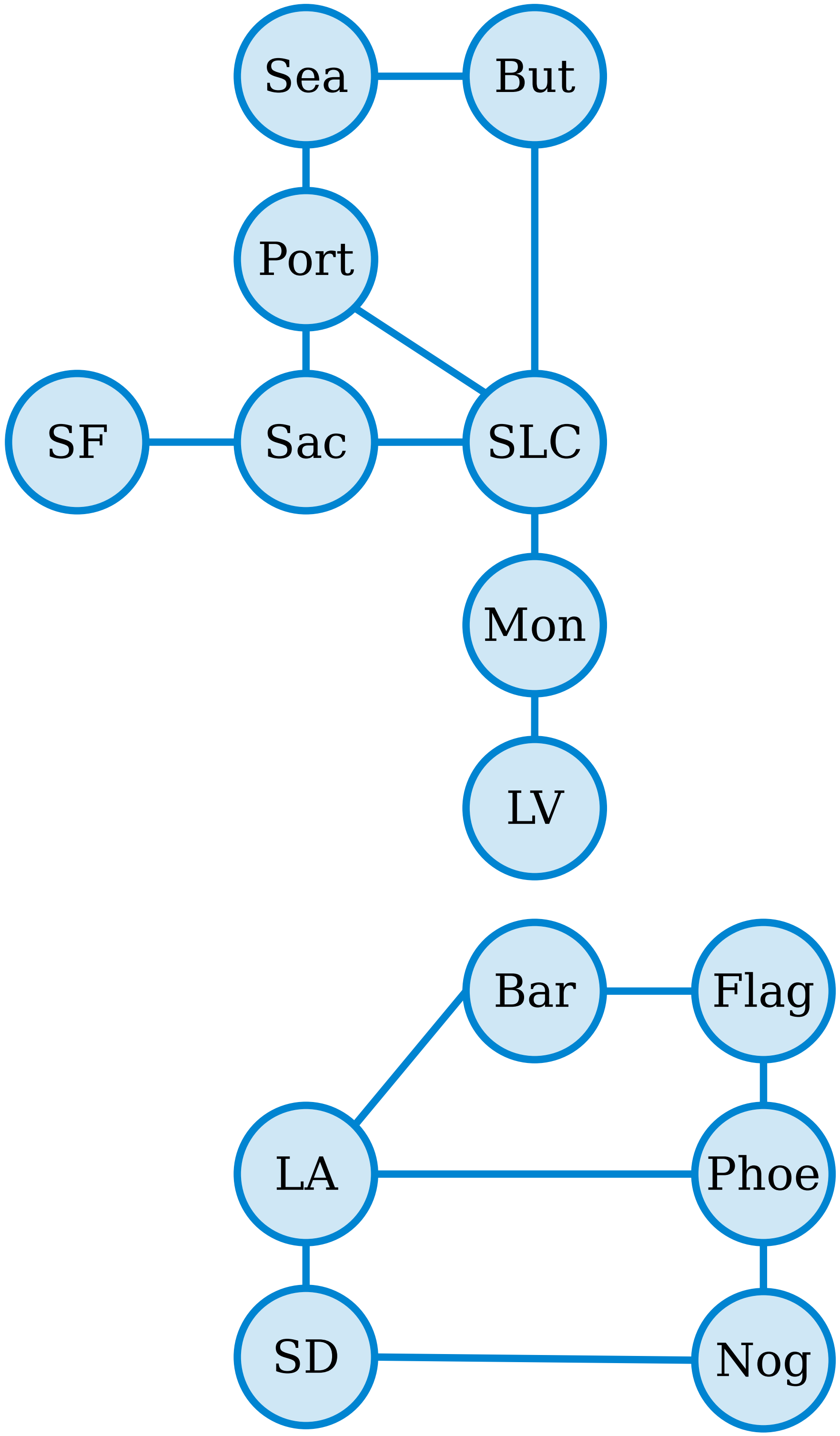


A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

A ***path*** in a graph is walk that does not repeat any nodes.

A node v is ***reachable*** from a node u if there is a path from u to v.

A graph G is called ***connected*** if all pairs of distinct nodes in G are reachable.



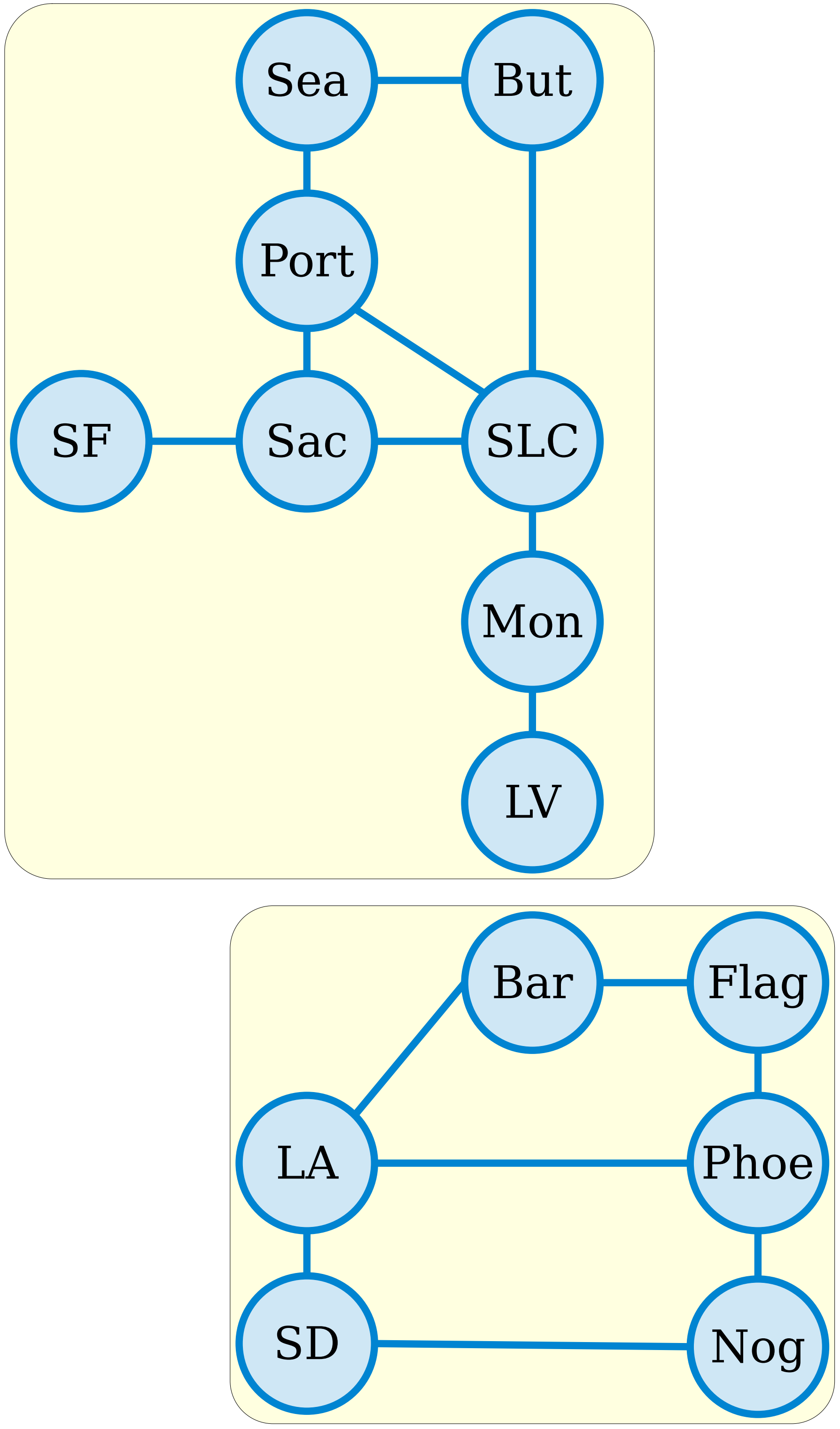
A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

A ***path*** in a graph is walk that does not repeat any nodes.

A node v is ***reachable*** from a node u if there is a path from u to v.

A graph G is called ***connected*** if all pairs of distinct nodes in G are reachable.

(This graph is not connected.)

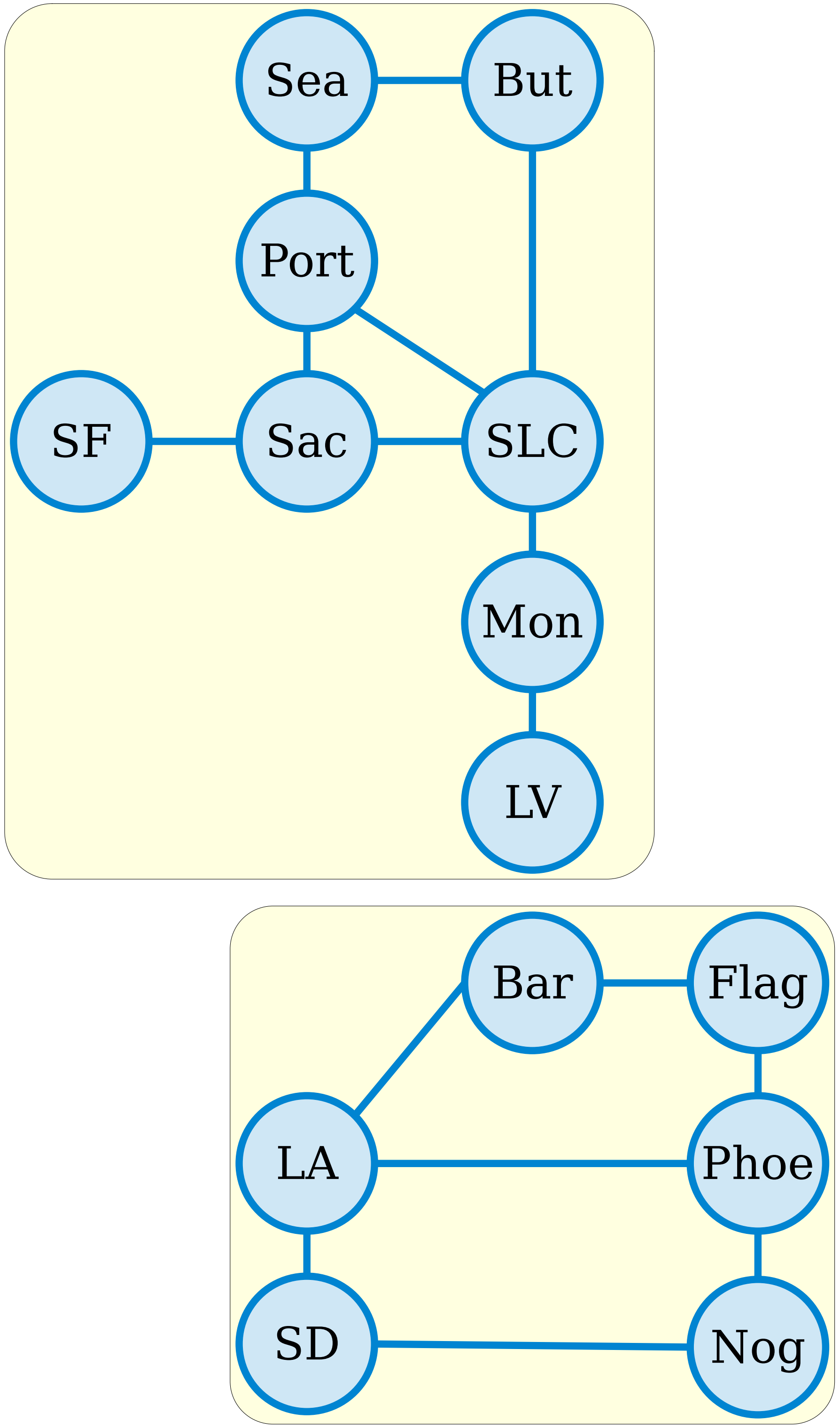


A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

A ***path*** in a graph is walk that does not repeat any nodes.

A node v is ***reachable*** from a node u if there is a path from u to v.

A graph G is called ***connected*** if all pairs of distinct nodes in G are reachable.



A ***walk*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

A ***path*** in a graph is walk that does not repeat any nodes.

A node v is ***reachable*** from a node u if there is a path from u to v.

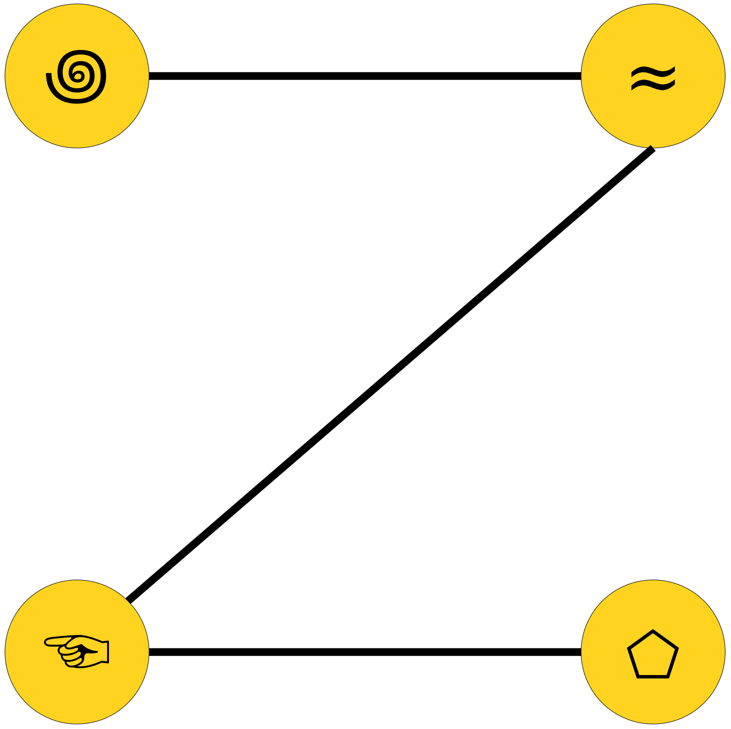
A graph G is called ***connected*** if all pairs of distinct nodes in G are reachable.

A ***connected component*** (or ***CC***) of G is a maximal set of mutually reachable nodes.

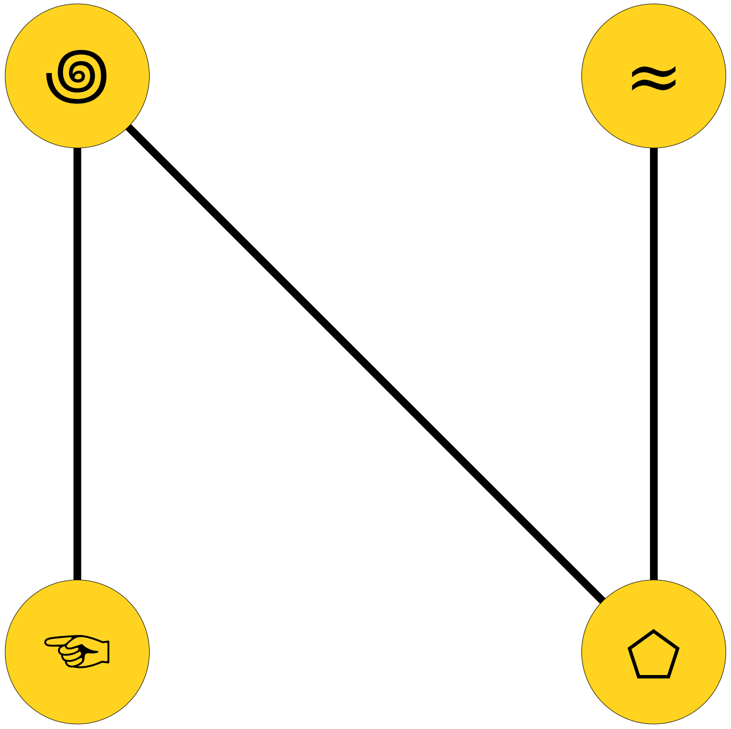
Fun Facts

- Here's a collection of useful facts about graphs that you can take as a given.
  - ***Theorem:*** If $G = (V, E)$ is a graph and $u, v \in V$, then there is a path from u to v if and only if there's a walk from u to v.
  - ***Theorem:*** If G is a graph and C is a cycle in G, then C's length is at least three and C contains at least three nodes.
  - ***Theorem:*** If $G = (V, E)$ is a graph, then every node in V belongs to exactly one connected component of G.
  - ***Theorem:*** If $G = (V, E)$ is a graph, then G is connected if and only if G has exactly one connected component.
- Looking for more practice working with formal definitions? Prove these results!

Graph Complements



Graph G



Graph G^c

Let $G = (V, E)$ be an undirected graph.
The ***complement of G*** is the graph $G^c = (V, E^c)$, where
$E^c = \{ \{u, v\} \mid u \in V, v \in V, u \neq v, \text{ and } \{u, v\} \notin E \}$

Theorem: For any graph $G = (V, E)$, at least one of G and G^c is connected.

Proving a Disjunction

- We need to prove the statement

  G is connected ∨ G^c is connected.

- Here's a neat observation.
  - If G is connected, we're done.
  - Otherwise, G isn't connected, and we have to prove that G^c is connected.
- We will therefore prove

  G is not connected → G^c is connected.

For any graph $G = (V, E)$,
at least one of G and G^c is connected.

Proving a Disjunction

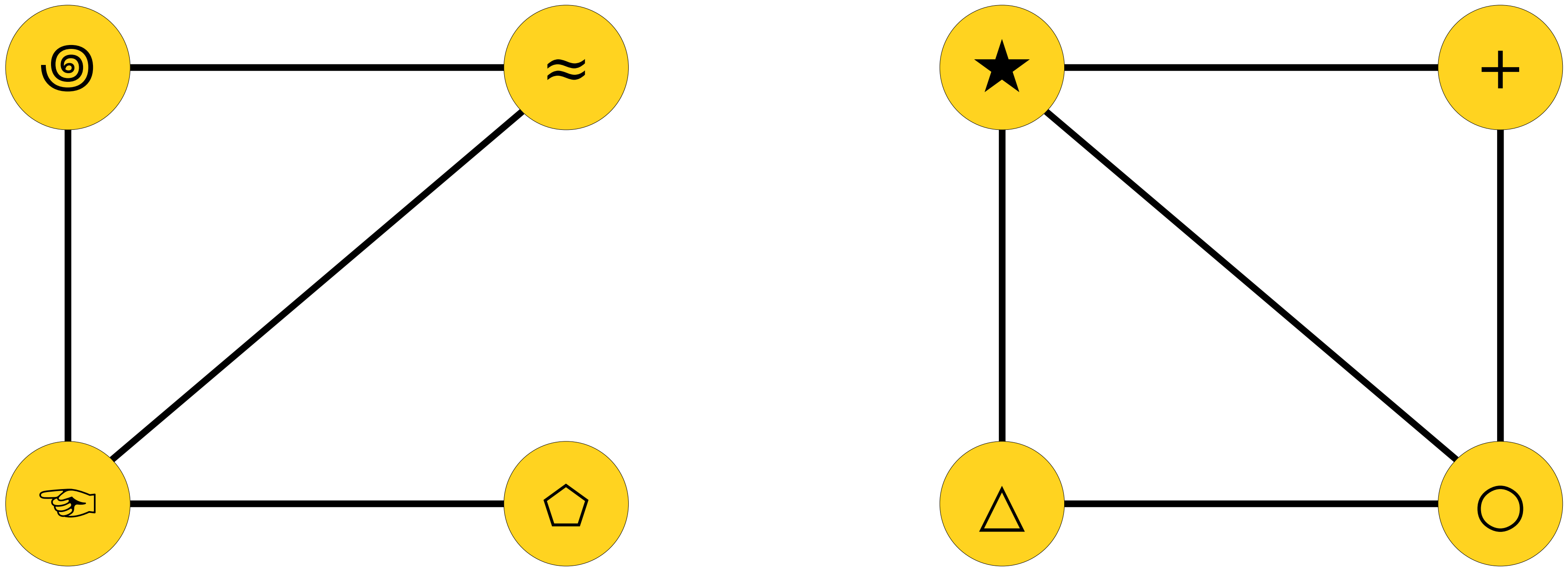
- We need to prove the statement

  G is connected   ∨ G^c is connected.

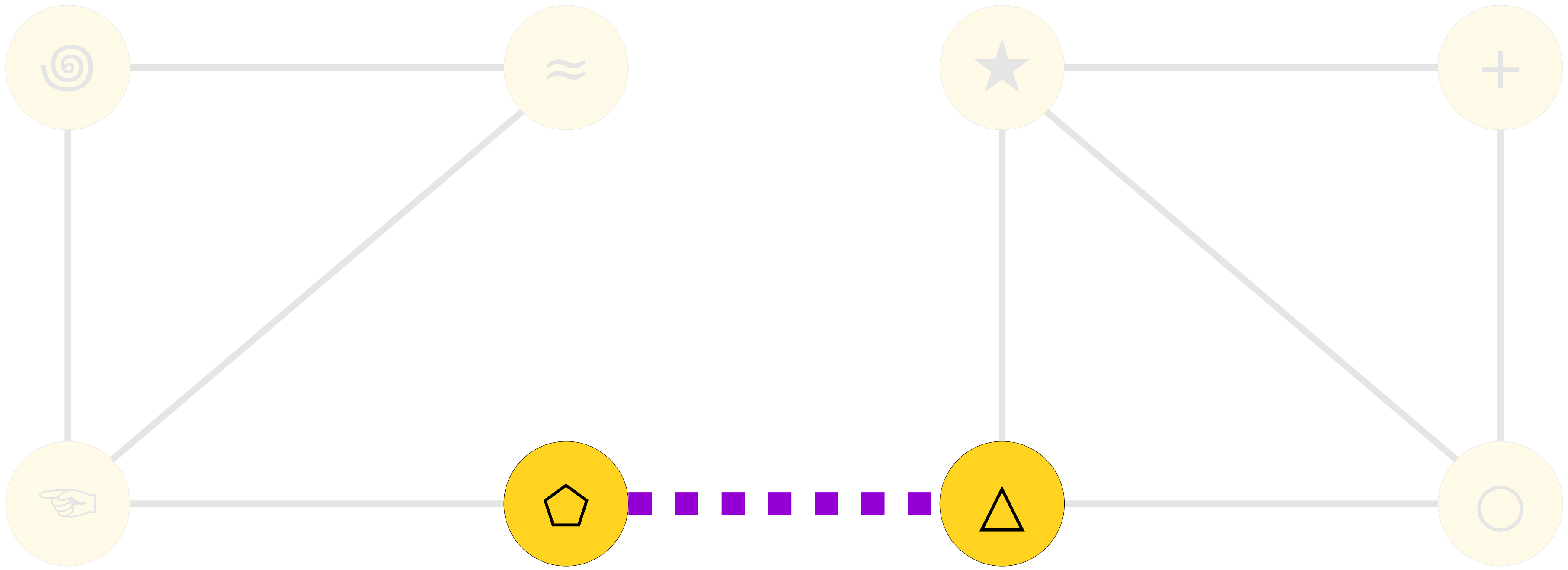
- Here's a neat observation.
  - If G is connected, we're done.
  - Otherwise, G isn't connected, and we have to prove that G^c is connected.
- We will therefore prove

  G is not connected   → G^c is connected.

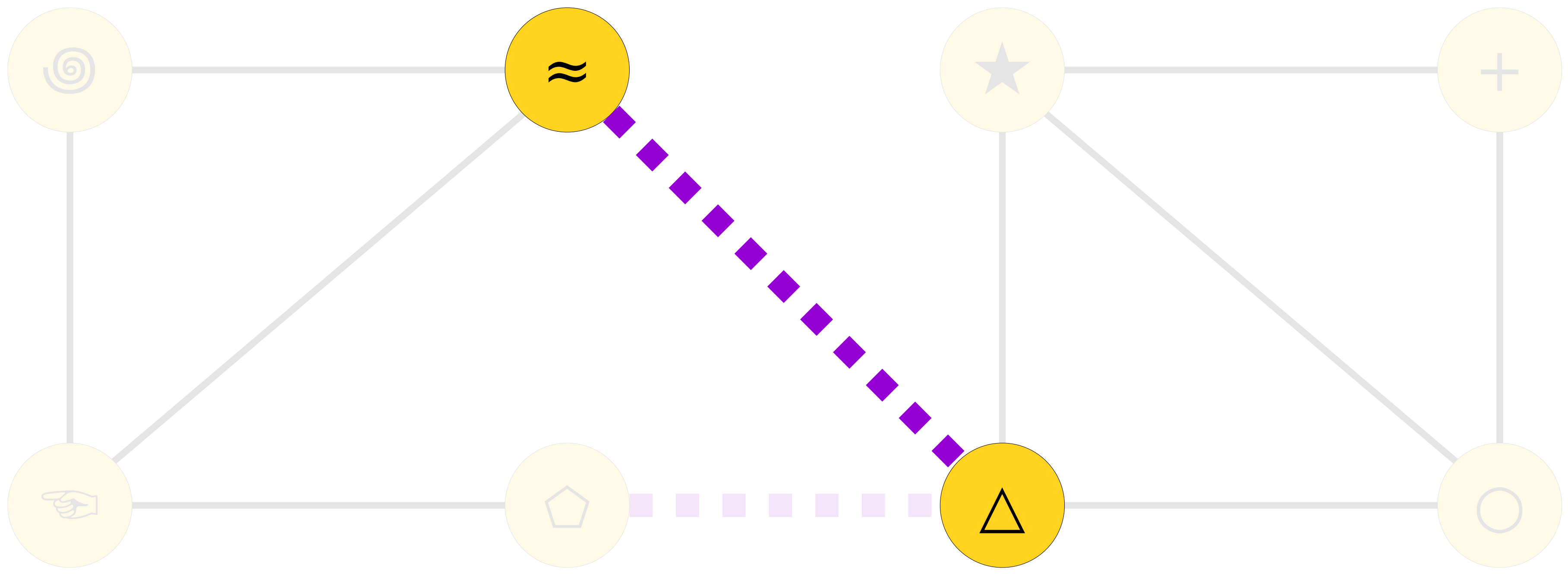
For any graph $G = (V, E)$,
if G is not connected, then G^c is connected.



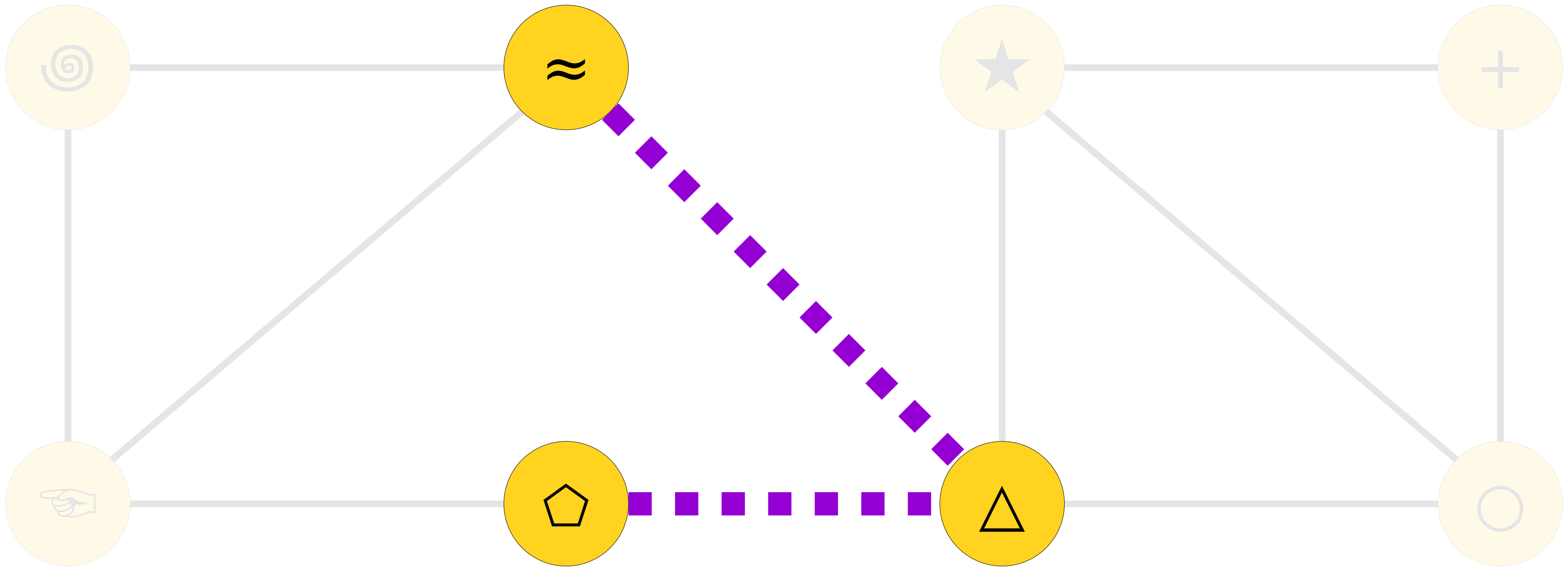
For any graph $G = (V, E)$,
if G is not connected, then G^c is connected.



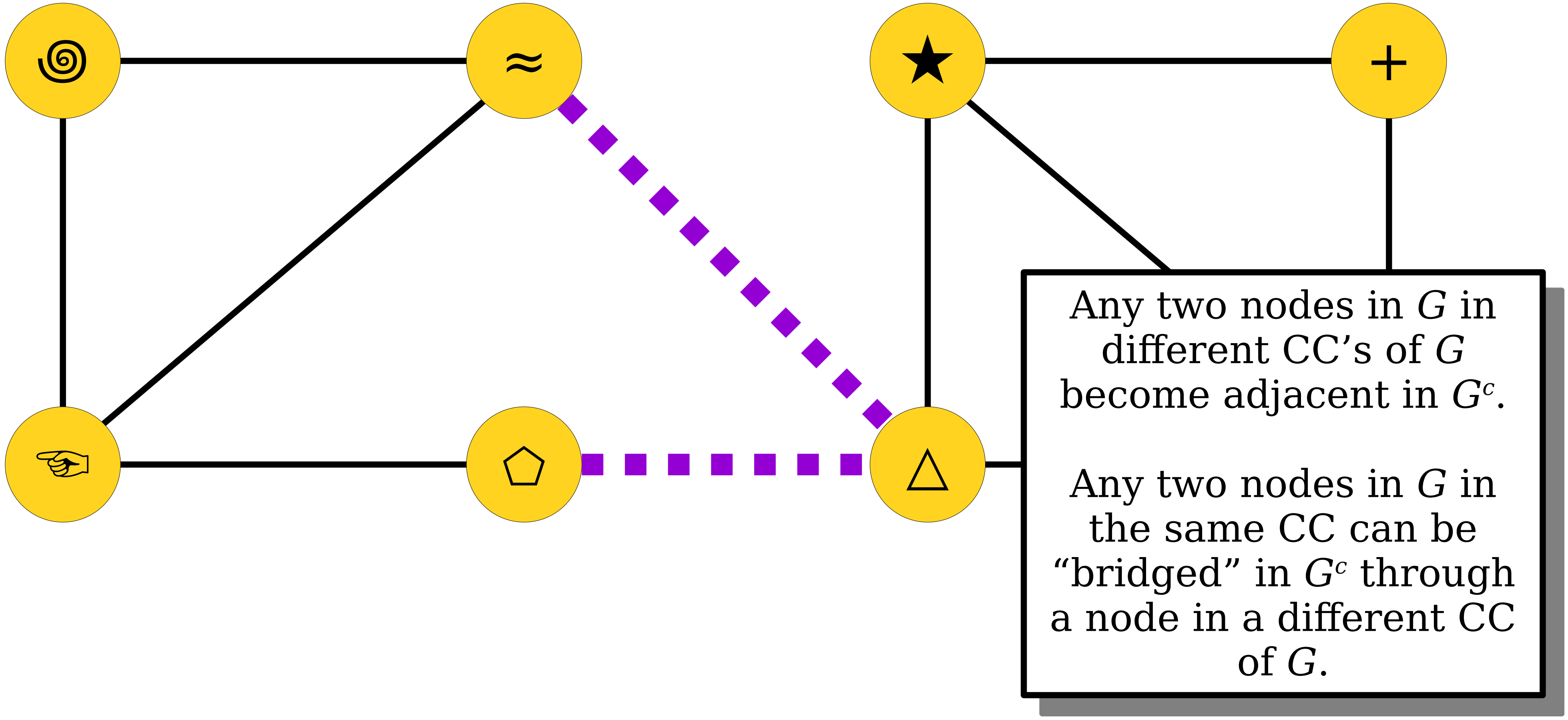
For any graph $G = (V, E)$,
if G is not connected, then G^c is connected.



For any graph $G = (V, E)$,
if G is not connected, then G^c is connected.



For any graph $G = (V, E)$,
if G is not connected, then G^c is connected.



Any two nodes in G in different CC's of G become adjacent in G^c.

Any two nodes in G in the same CC can be "bridged" in G^c through a node in a different CC of G.

For any graph $G = (V, E)$, if G is not connected, then G^c is connected.

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Proof:

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Proof: Let $G = (V, E)$ be an arbitrary graph and assume G is not connected.

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Proof: Let $G = (V, E)$ be an arbitrary graph and assume G is not connected. We need to show that $G^c = (V, E^c)$ is connected.

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Proof: Let $G = (V, E)$ be an arbitrary graph and assume G is not connected. We need to show that $G^c = (V, E^c)$ is connected. To do so, consider any two distinct nodes $u, v \in V$.

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Proof: Let $G = (V, E)$ be an arbitrary graph and assume G is not connected. We need to show that $G^c = (V, E^c)$ is connected. To do so, consider any two distinct nodes $u, v \in V$. We need to show that there is a path from u to v in G^c.

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Proof: Let $G = (V, E)$ be an arbitrary graph and assume G is not connected. We need to show that $G^c = (V, E^c)$ is connected. To do so, consider any two distinct nodes $u, v \in V$. We need to show that there is a path from u to v in G^c. We consider two cases:

Case 1: u and v are in different connected components of G.

Case 2: u and v are in the same connected component of G.

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Proof: Let $G = (V, E)$ be an arbitrary graph and assume G is not connected. We need to show that $G^c = (V, E^c)$ is connected. To do so, consider any two distinct nodes $u, v \in V$. We need to show that there is a path from u to v in G^c. We consider two cases:

Case 1: u and v are in different connected components of G. This means that $\{u, v\} \notin E$, since otherwise the path u, v would make u reachable from v and they'd be in the same connected component of G.

Case 2: u and v are in the same connected component of G.

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Proof: Let $G = (V, E)$ be an arbitrary graph and assume G is not connected. We need to show that $G^c = (V, E^c)$ is connected. To do so, consider any two distinct nodes $u, v \in V$. We need to show that there is a path from u to v in G^c. We consider two cases:

Case 1: u and v are in different connected components of G. This means that $\{u, v\} \notin E$, since otherwise the path u, v would make u reachable from v and they'd be in the same connected component of G. Therefore, we see that $\{u, v\} \in E^c$, and so there is a path (namely, u, v) from u to v in G^c.

Case 2: u and v are in the same connected component of G.

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Proof: Let $G = (V, E)$ be an arbitrary graph and assume G is not connected. We need to show that $G^c = (V, E^c)$ is connected. To do so, consider any two distinct nodes $u, v \in V$. We need to show that there is a path from u to v in G^c. We consider two cases:

Case 1: u and v are in different connected components of G. This means that $\{u, v\} \notin E$, since otherwise the path u, v would make u reachable from v and they'd be in the same connected component of G. Therefore, we see that $\{u, v\} \in E^c$, and so there is a path (namely, u, v) from u to v in G^c.

Case 2: u and v are in the same connected component of G. Since G is not connected, there are at least two connected components of G.

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Proof: Let $G = (V, E)$ be an arbitrary graph and assume G is not connected. We need to show that $G^c = (V, E^c)$ is connected. To do so, consider any two distinct nodes $u, v \in V$. We need to show that there is a path from u to v in G^c. We consider two cases:

Case 1: u and v are in different connected components of G. This means that $\{u, v\} \notin E$, since otherwise the path u, v would make u reachable from v and they'd be in the same connected component of G. Therefore, we see that $\{u, v\} \in E^c$, and so there is a path (namely, u, v) from u to v in G^c.

Case 2: u and v are in the same connected component of G. Since G is not connected, there are at least two connected components of G. Pick any node z that belongs to a different connected component of G than u and v.

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Proof: Let $G = (V, E)$ be an arbitrary graph and assume G is not connected. We need to show that $G^c = (V, E^c)$ is connected. To do so, consider any two distinct nodes $u, v \in V$. We need to show that there is a path from u to v in G^c. We consider two cases:

Case 1: u and v are in different connected components of G. This means that $\{u, v\} \notin E$, since otherwise the path u, v would make u reachable from v and they'd be in the same connected component of G. Therefore, we see that $\{u, v\} \in E^c$, and so there is a path (namely, u, v) from u to v in G^c.

Case 2: u and v are in the same connected component of G. Since G is not connected, there are at least two connected components of G. Pick any node z that belongs to a different connected component of G than u and v. Then by the reasoning from Case 1 we know that $\{u, z\} \in E^c$ and $\{z, v\} \in E^c$.

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Proof: Let $G = (V, E)$ be an arbitrary graph and assume G is not connected. We need to show that $G^c = (V, E^c)$ is connected. To do so, consider any two distinct nodes $u, v \in V$. We need to show that there is a path from u to v in G^c. We consider two cases:

Case 1: u and v are in different connected components of G. This means that $\{u, v\} \notin E$, since otherwise the path u, v would make u reachable from v and they'd be in the same connected component of G. Therefore, we see that $\{u, v\} \in E^c$, and so there is a path (namely, u, v) from u to v in G^c.

Case 2: u and v are in the same connected component of G. Since G is not connected, there are at least two connected components of G. Pick any node z that belongs to a different connected component of G than u and v. Then by the reasoning from Case 1 we know that $\{u, z\} \in E^c$ and $\{z, v\} \in E^c$. This gives a path u, z, v in G^c from u to v.

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Proof: Let $G = (V, E)$ be an arbitrary graph and assume G is not connected. We need to show that $G^c = (V, E^c)$ is connected. To do so, consider any two distinct nodes $u, v \in V$. We need to show that there is a path from u to v in G^c. We consider two cases:

Case 1: u and v are in different connected components of G. This means that $\{u, v\} \notin E$, since otherwise the path u, v would make u reachable from v and they'd be in the same connected component of G. Therefore, we see that $\{u, v\} \in E^c$, and so there is a path (namely, u, v) from u to v in G^c.

Case 2: u and v are in the same connected component of G. Since G is not connected, there are at least two connected components of G. Pick any node z that belongs to a different connected component of G than u and v. Then by the reasoning from Case 1 we know that $\{u, z\} \in E^c$ and $\{z, v\} \in E^c$. This gives a path u, z, v in G^c from u to v.

In either case, we find a path from u to v in G^c, as required.

Theorem: If $G = (V, E)$ is a graph, then at least one of G and G^c is connected.

Proof: Let $G = (V, E)$ be an arbitrary graph and assume G is not connected. We need to show that $G^c = (V, E^c)$ is connected. To do so, consider any two distinct nodes $u, v \in V$. We need to show that there is a path from u to v in G^c. We consider two cases:

Case 1: u and v are in different connected components of G. This means that $\{u, v\} \notin E$, since otherwise the path u, v would make u reachable from v and they'd be in the same connected component of G. Therefore, we see that $\{u, v\} \in E^c$, and so there is a path (namely, u, v) from u to v in G^c.

Case 2: u and v are in the same connected component of G. Since G is not connected, there are at least two connected components of G. Pick any node z that belongs to a different connected component of G than u and v. Then by the reasoning from Case 1 we know that $\{u, z\} \in E^c$ and $\{z, v\} \in E^c$. This gives a path u, z, v in G^c from u to v.

In either case, we find a path from u to v in G^c, as required. ■

Time-Out for Announcements!

Problem Set Two Graded

- Your diligent and hardworking TAs have just finished grading PS2. Grades and feedback are now available on Gradescope.
  - 75th Percentile: 69 / 74 (93%)
  - 50th Percentile: 67 / 74 (91%)
  - 25th Percentile: 61 / 74 (82%)
- As always, ***please review your feedback!*** Knowing where to improve is more important than just seeing a raw score.
- Did we make a mistake? Regrades will open up on Friday and are due by next Wednesday.

What’s On Deck

- The first midterm goes out on Friday at 2:30PM Pacific. It comes due on Sunday at 2:30PM Pacific.
    - The exam must be completed individually.
    - It’s open-book, open-note, and closed-other-humans.
    - It covers PS1 – PS2 and L00 – L05. Functions and onward aren’t tested (yet).
- We will have class on Friday. We’re giving you next Monday off.
- PS4 will go out on Friday at 2:30PM as usual, with a due date of next Friday at 2:30PM as usual. It’s designed to be shorter than normal, since we don’t expect you to start working on it / look at it until Monday.

Preparing for the Exam

- The best way to prepare for the exam is to
  - ***work on PS3***, which covers proofs, first-order logic, and the like, and
  - ***review your feedback*** on PS1 and PS2 so you know what to keep an eye out for as you complete PS3.
- If you want more targeted practice with any topics we've covered this quarter, there are eighteen extra practice problems available on the course website.
- Best of luck – ***you can do this!***

Your Questions

“Advice for dealing with stress / burnout / exhaustion?” “Advice for time management and finding balance? Have felt overwhelmed with work recently and don't know how to catch up.”

For starters – I'm sorry to hear you're feeling this. If you'd like to chat about anything, feel free to ping me.

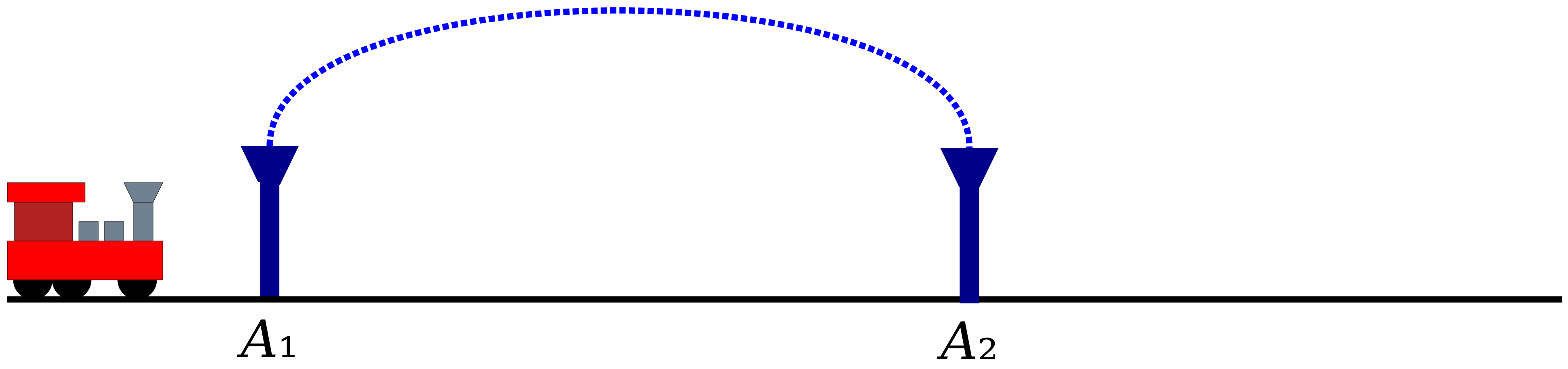
What do you do to recharge? Some people I know like to meditate, others exercise, etc. Having tools like these you can deploy can make a world of difference. And make sure you're doing proper biological care and maintenance. Are you eating enough? Sleeping enough? Getting proper exercise? If not, it is well worth doing so. It is easy to lose track of just how important these are.

Time management is a skill. If you're overloaded, identify ways to reduce your commitments. That could be something like dropping a class, or it could mean taking on a less active role in a student group. It could also mean doing a passable job with something rather than knocking it out of the park.

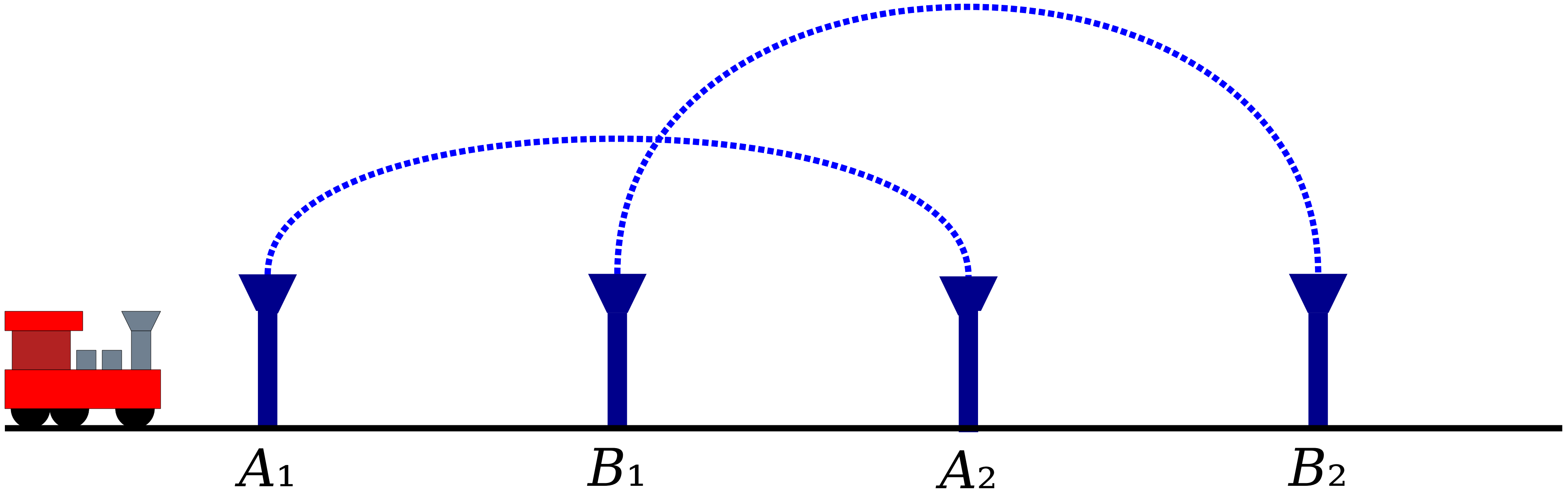
And if you've done all this and you're still feeling overwhelmed, reach out to someone who can give you personalized advice, whether that's through informal mentorship channels or more structured counseling. It is perfectly normal to get help this way – that's why these resources exist!

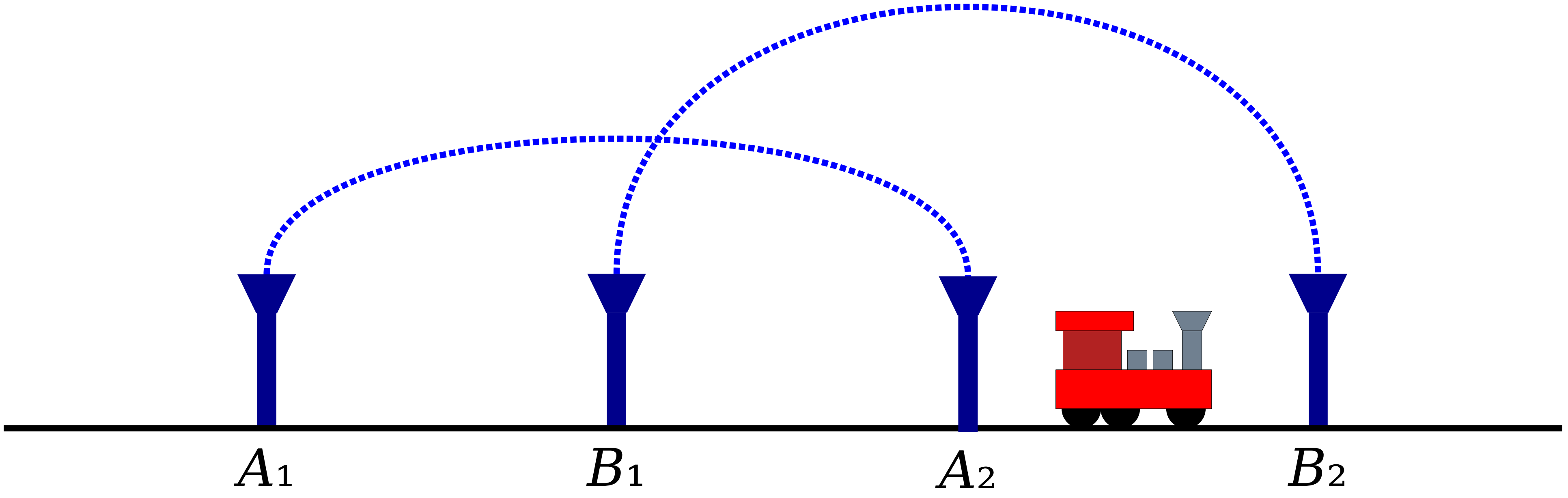
Back to CS103!

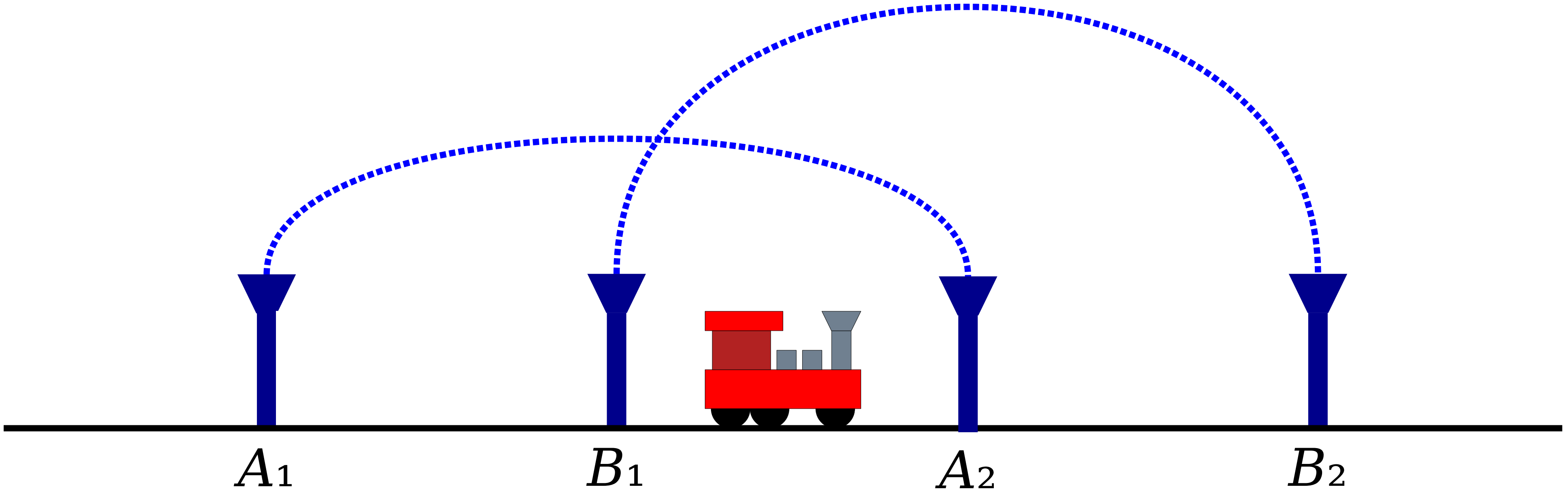
The Teleported Train Problem

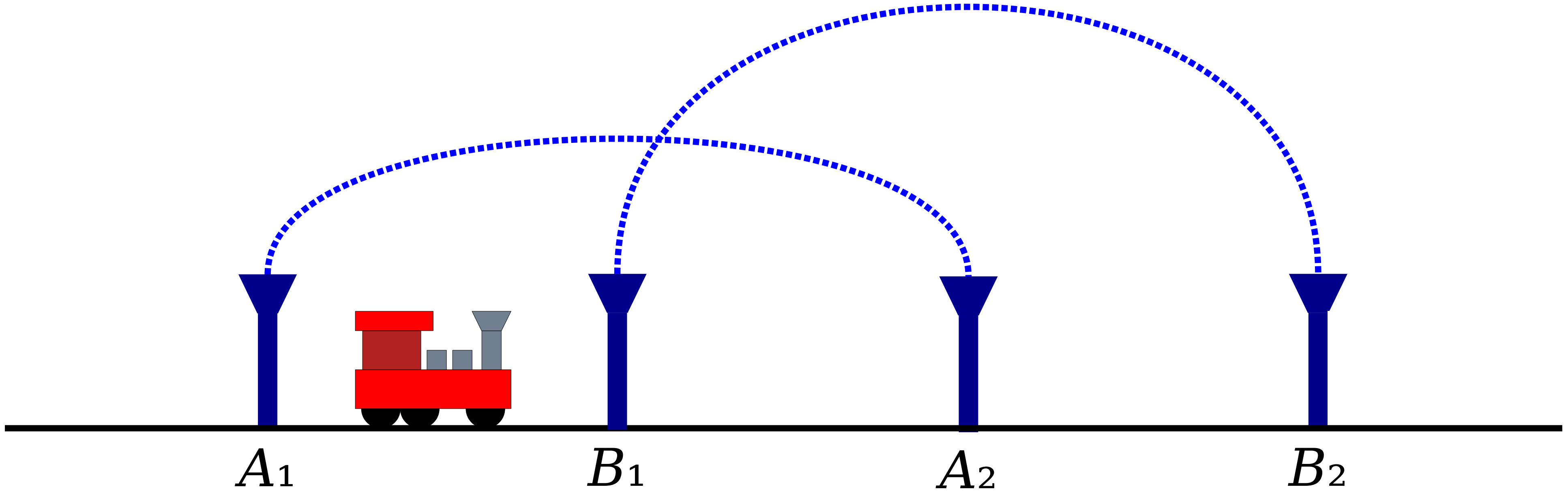


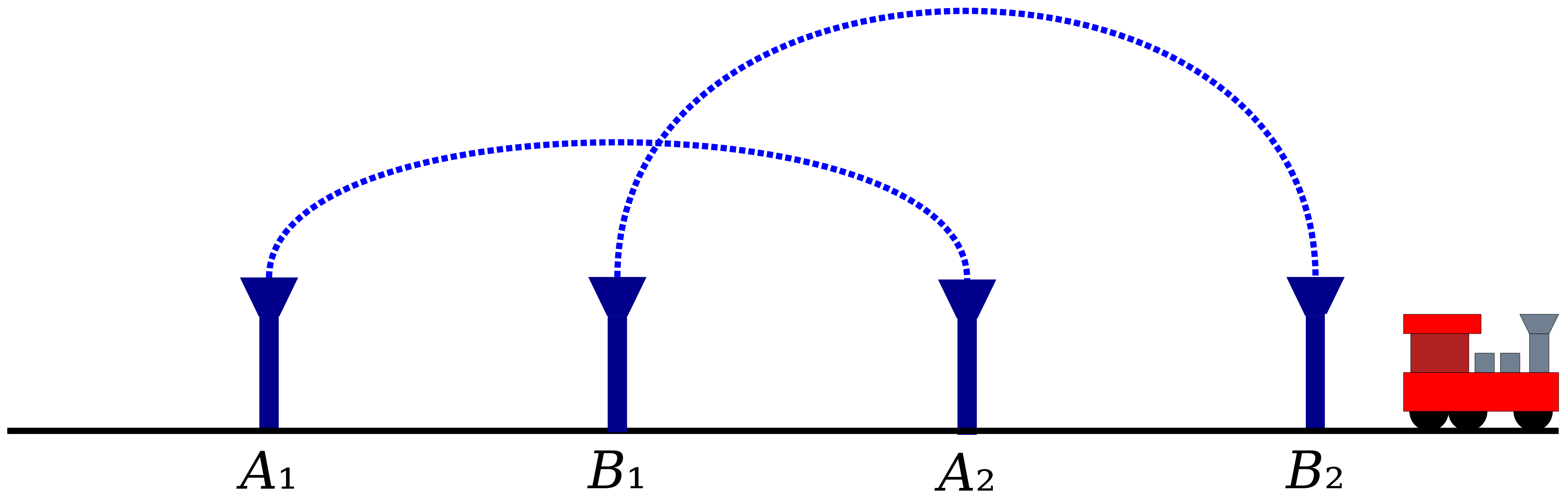
These are ***teleporters***. Anything entering a teleporter from the left side emerges from the right side of the paired teleporter.


A_1
B_1
A_2
B_2

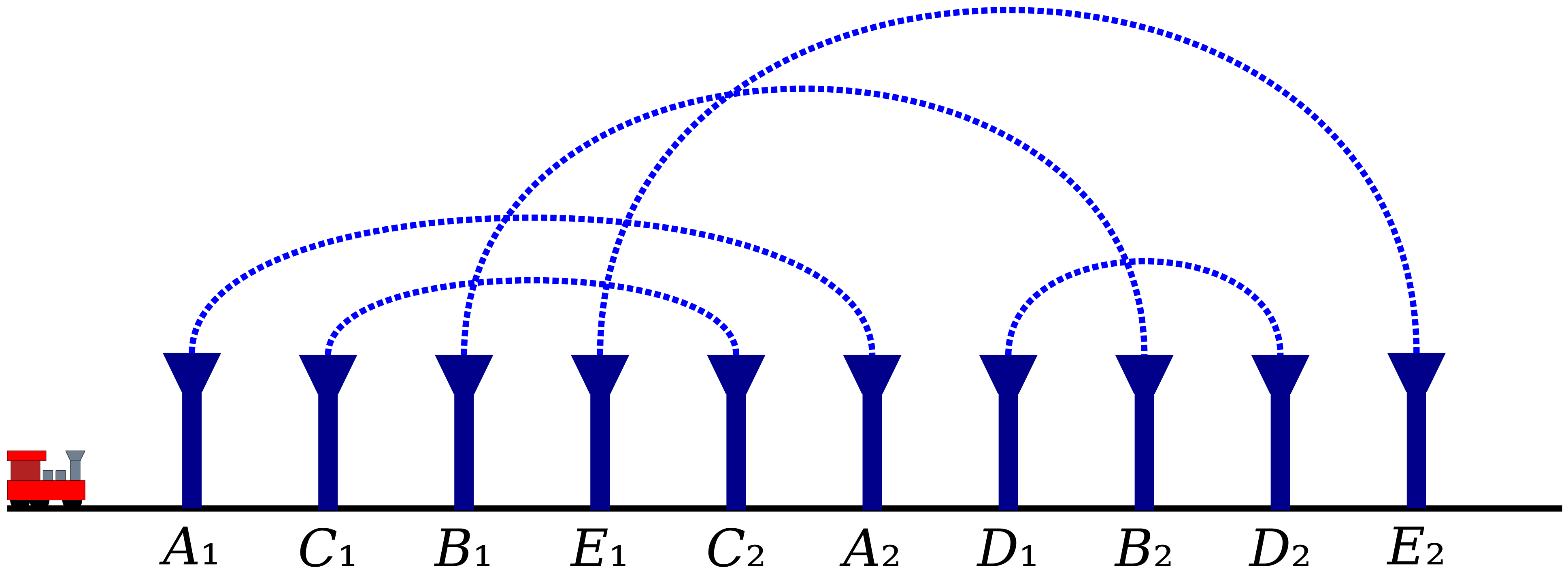

A_1
B_1
A_2
B_2


A_1
B_1
A_2
B_2

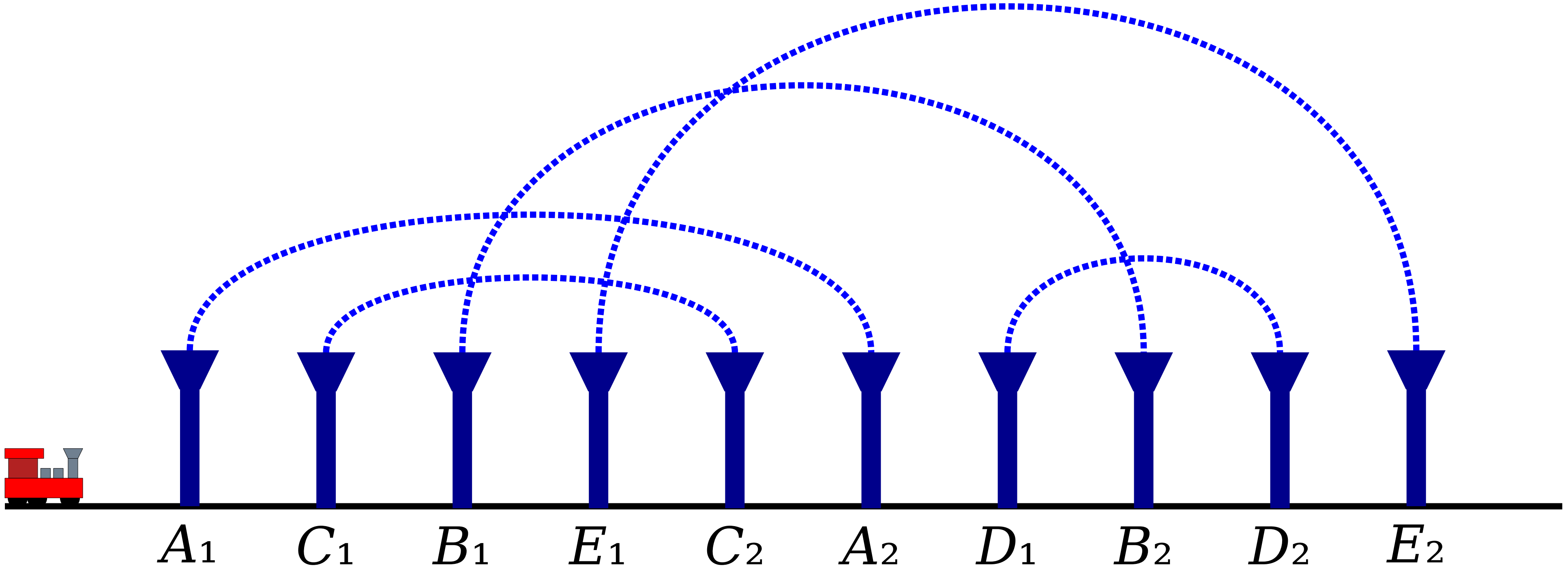

A_1
B_1
A_2
B_2

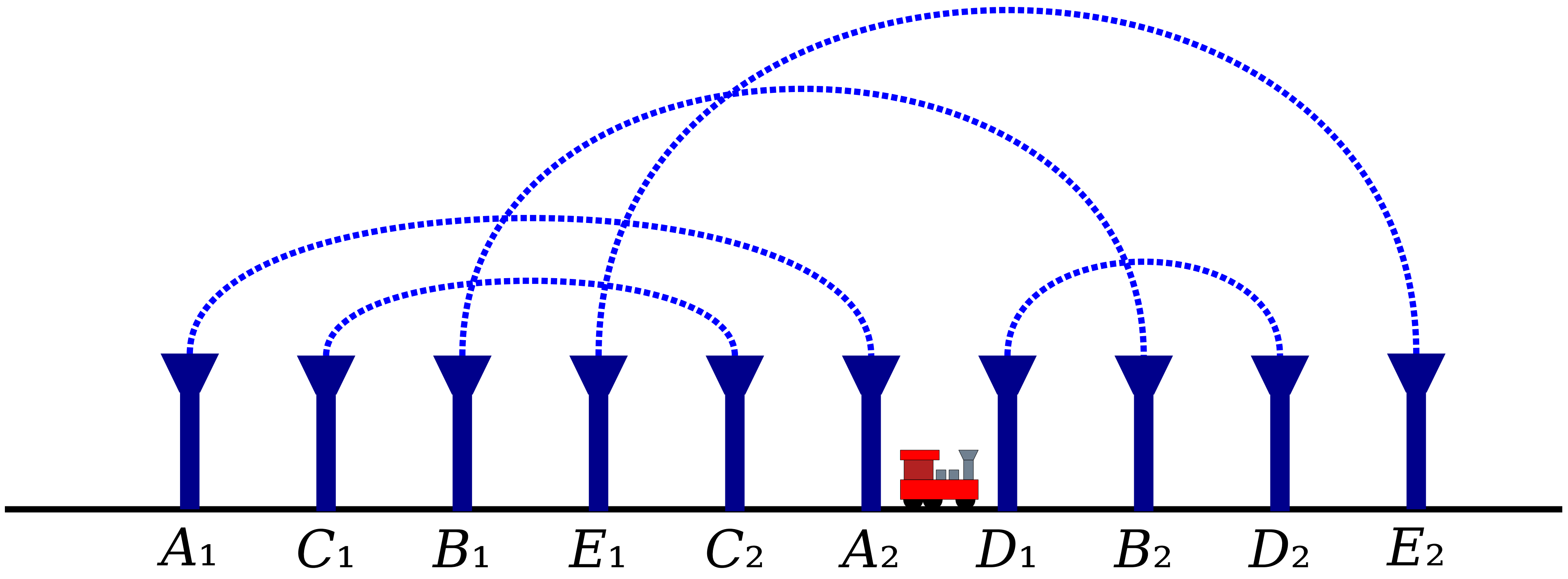


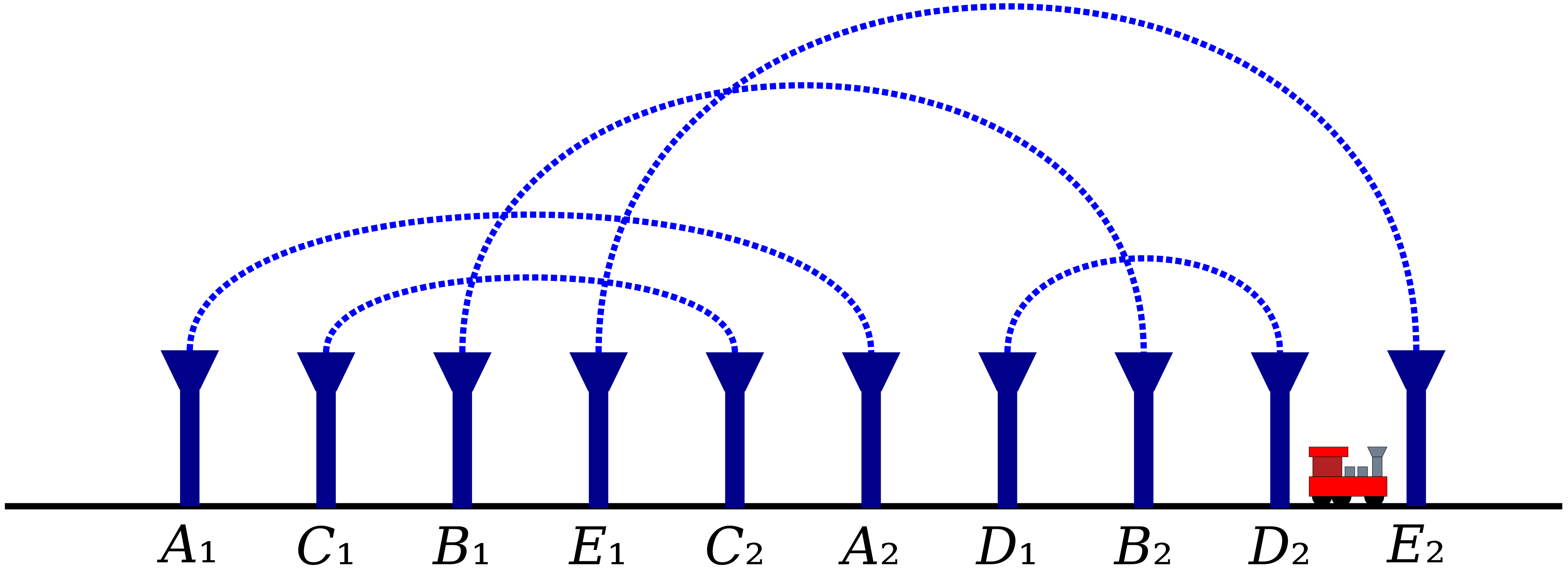
It took a while, but eventually the train reached the end of the track.

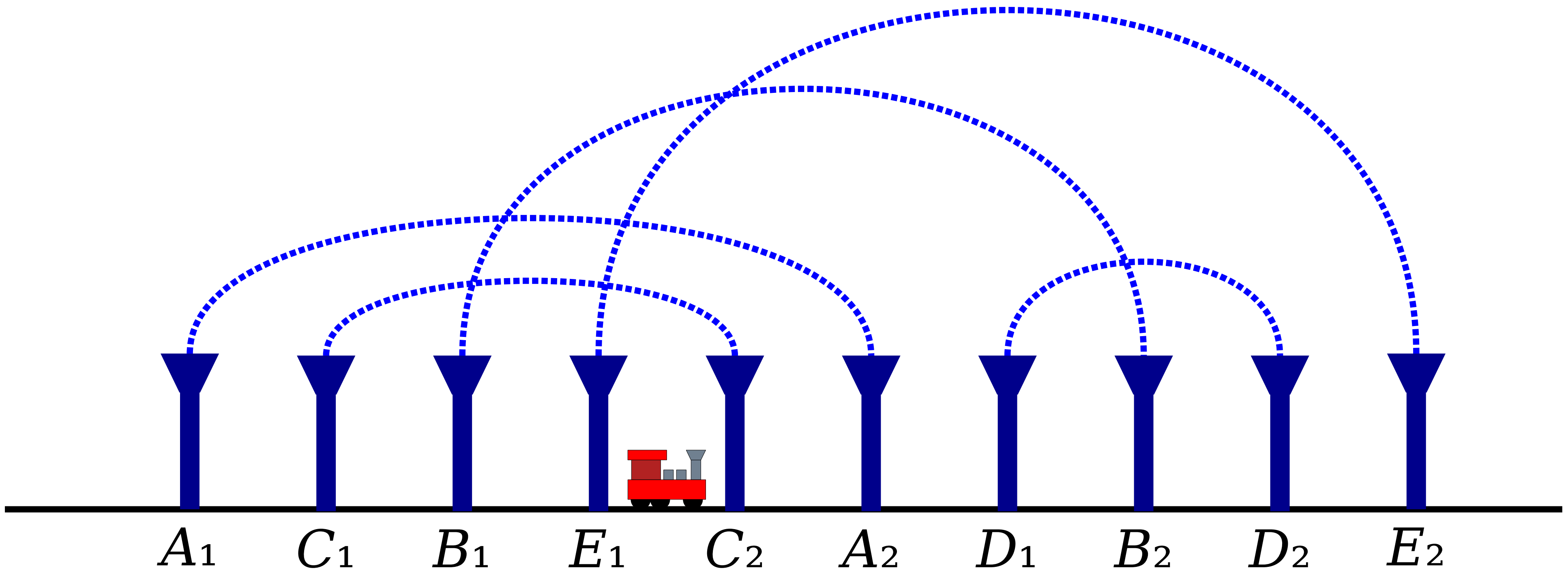


Will the train reach the end of the track? Or will it get stuck in a loop?

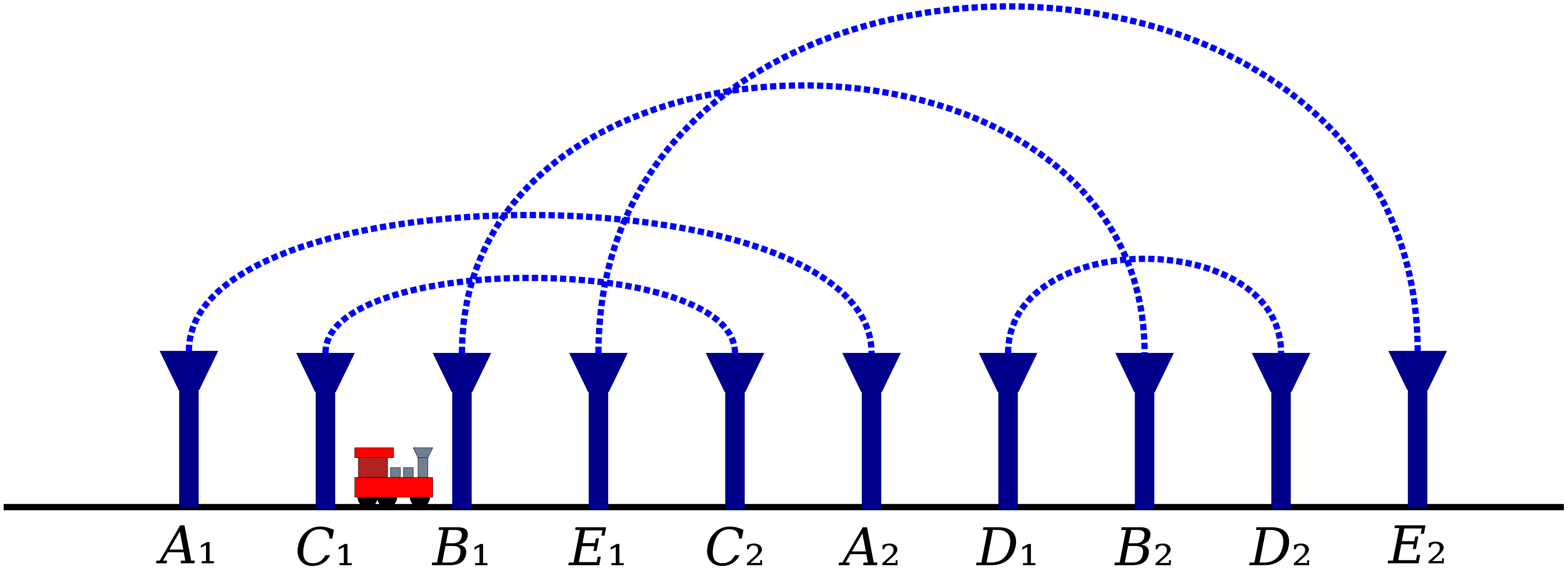

A_1
C_1
B_1
E_1
C_2
A_2
D_1
B_2
D_2
E_2

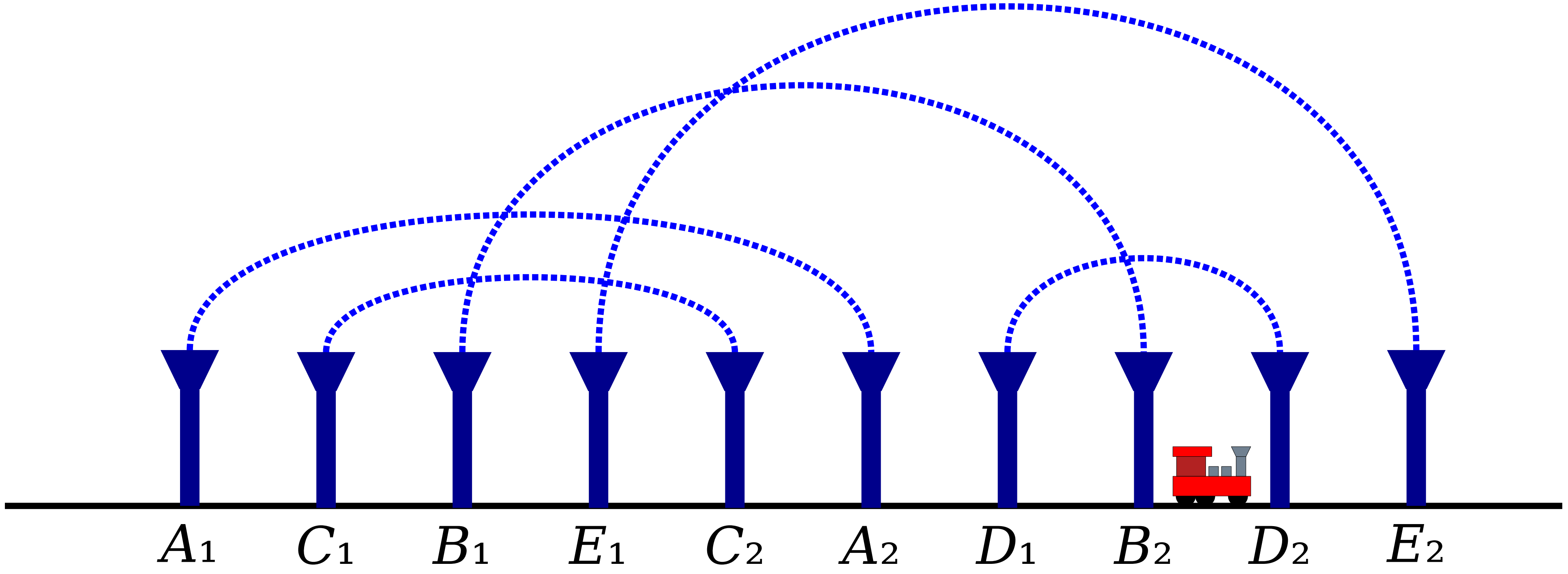

A_1
C_1
B_1
E_1
C_2
A_2
D_1
B_2
D_2
E_2

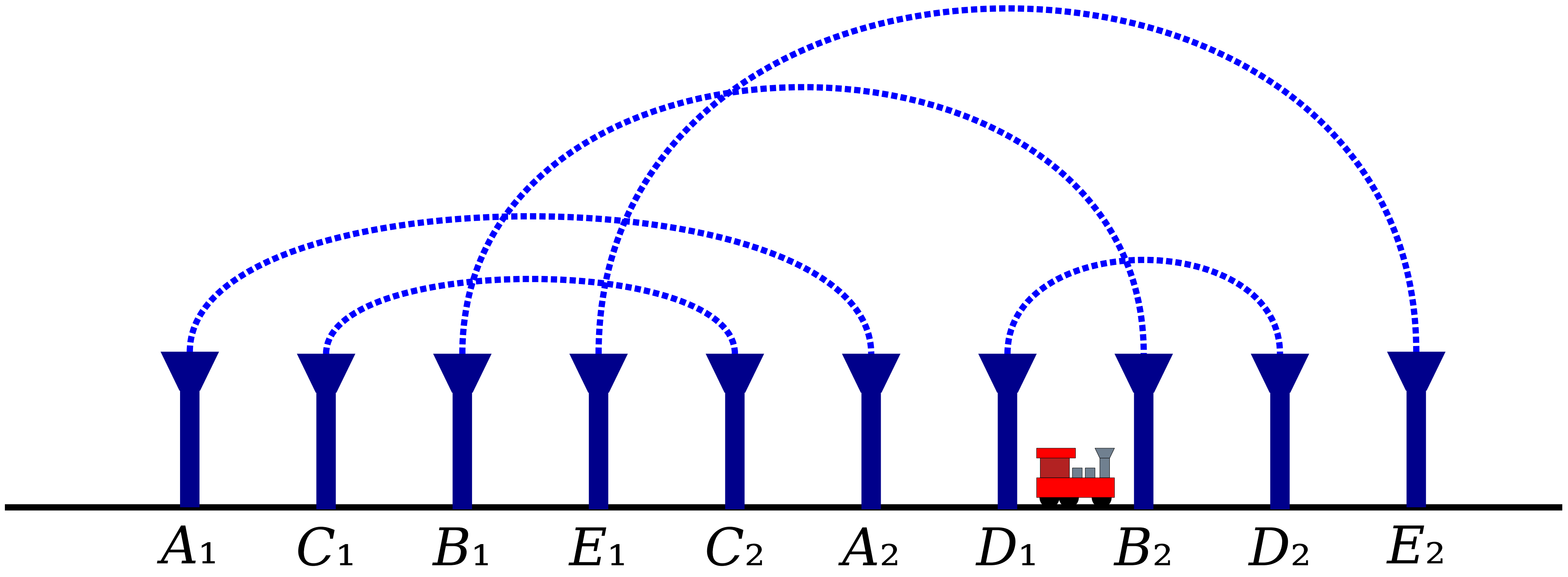

A_1
C_1
B_1
E_1
C_2
A_2
D_1
B_2
D_2
E_2

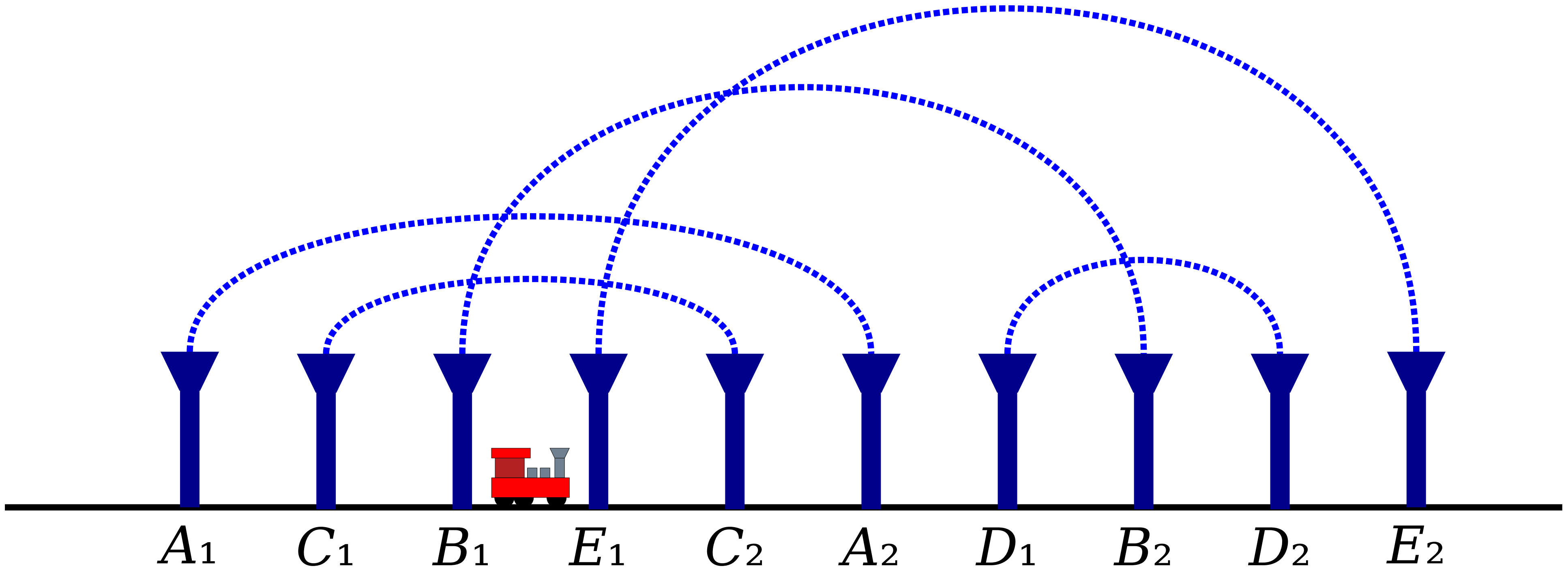


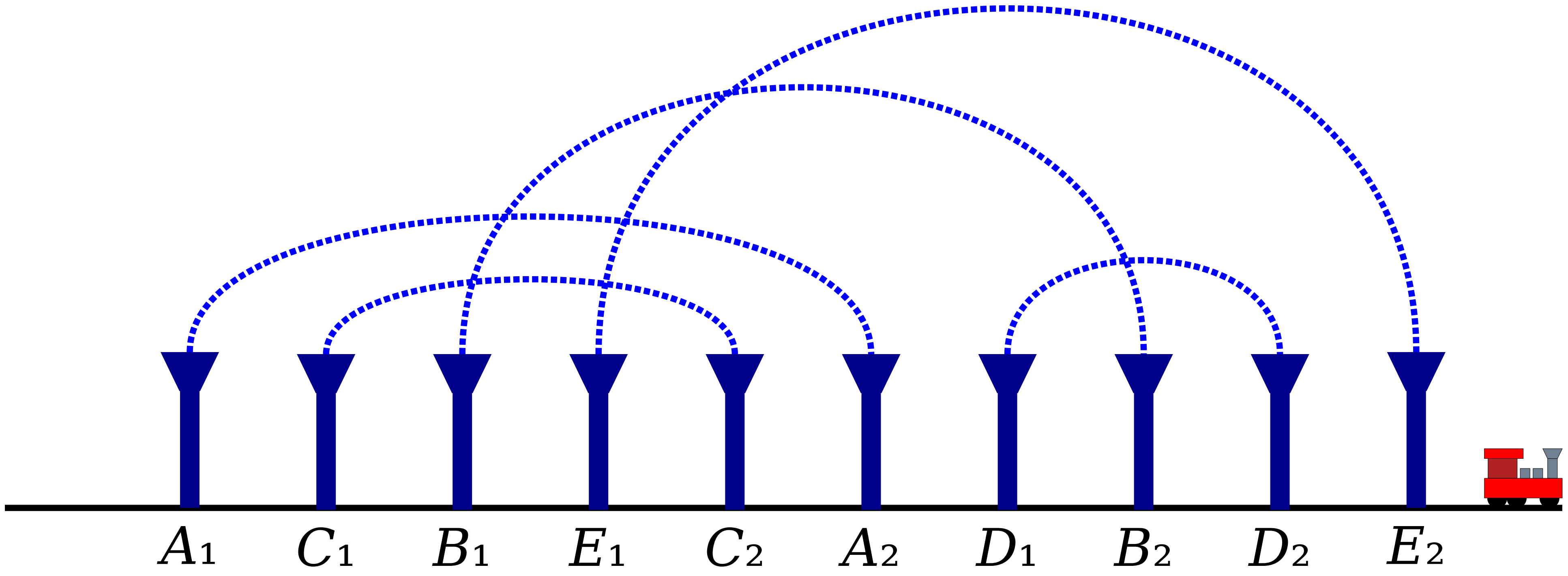
A_1
C_1
B_1
E_1
C_2
A_2
D_1
B_2
D_2
E_2

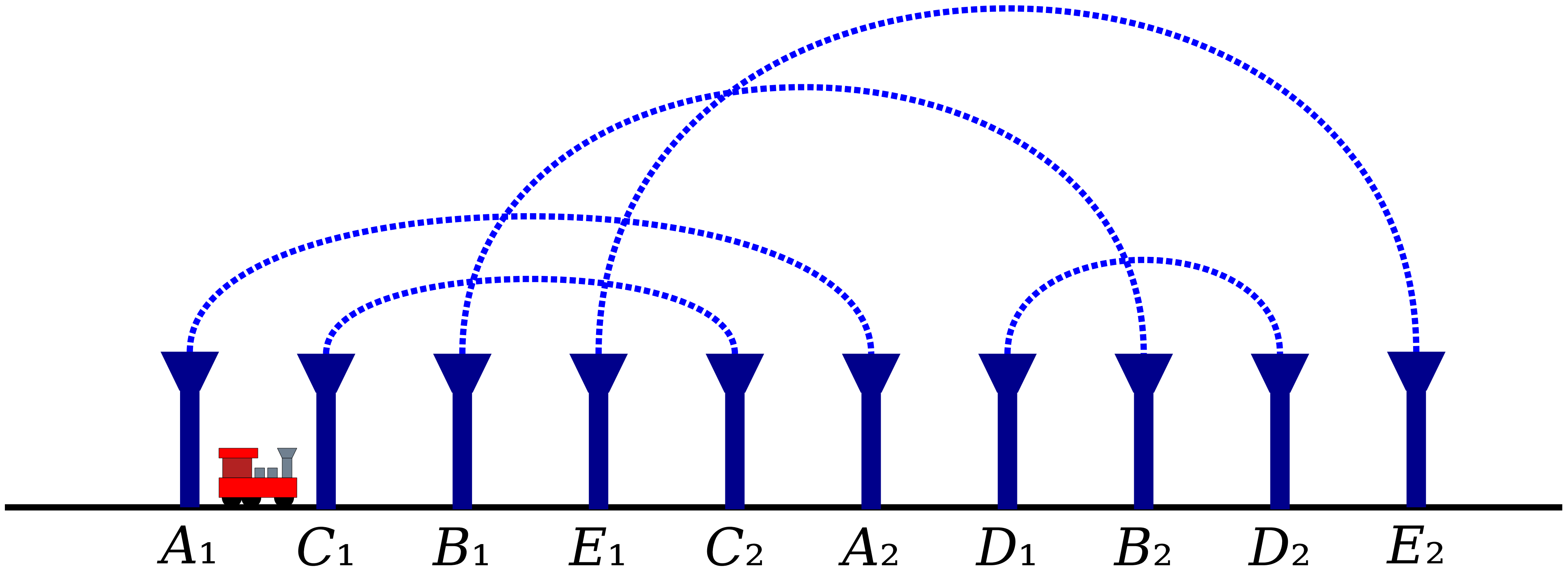

A_1
C_1
B_1
E_1
C_2
A_2
D_1
B_2
D_2
E_2

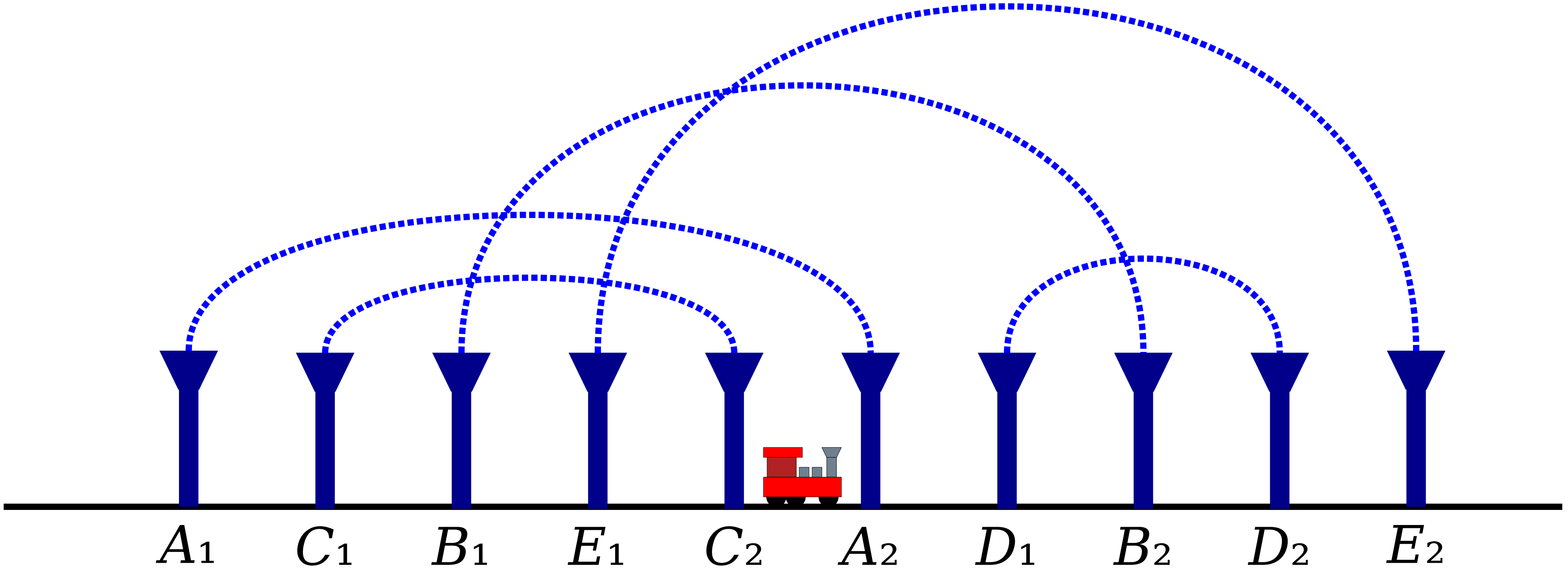

A_1
C_1
B_1
E_1
C_2
A_2
D_1
B_2
D_2
E_2

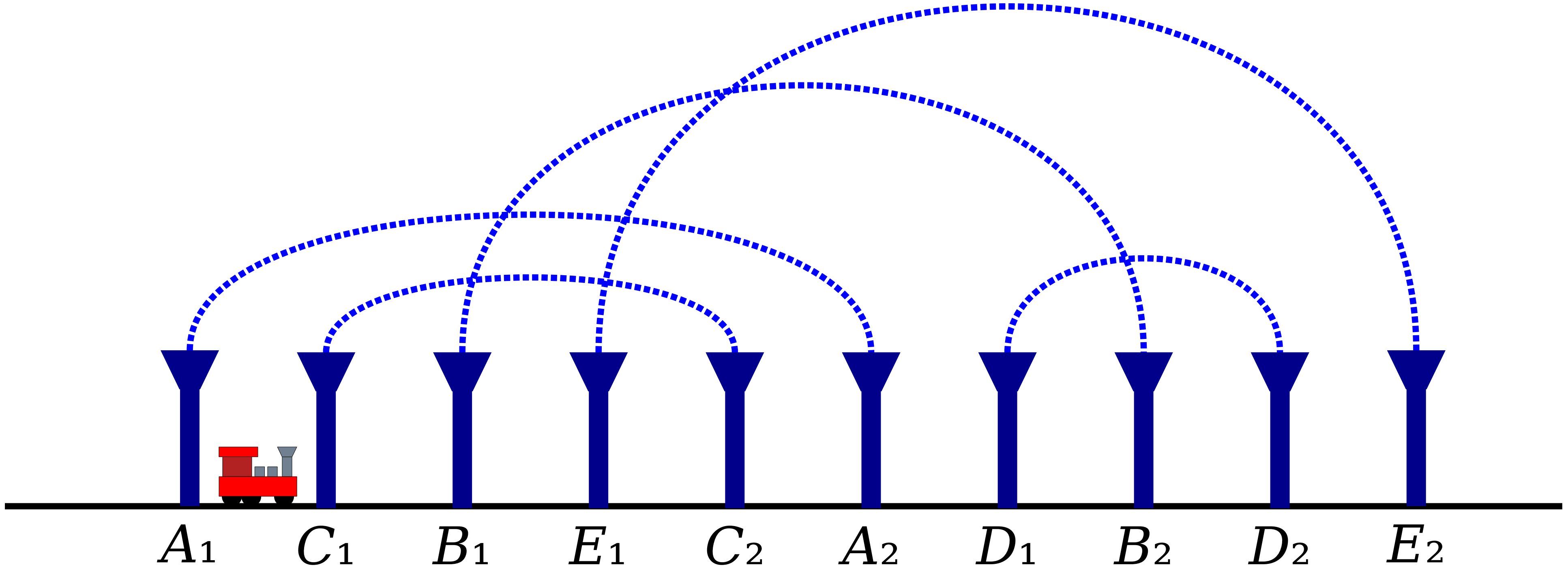

A_1
C_1
B_1
E_1
C_2
A_2
D_1
B_2
D_2
E_2

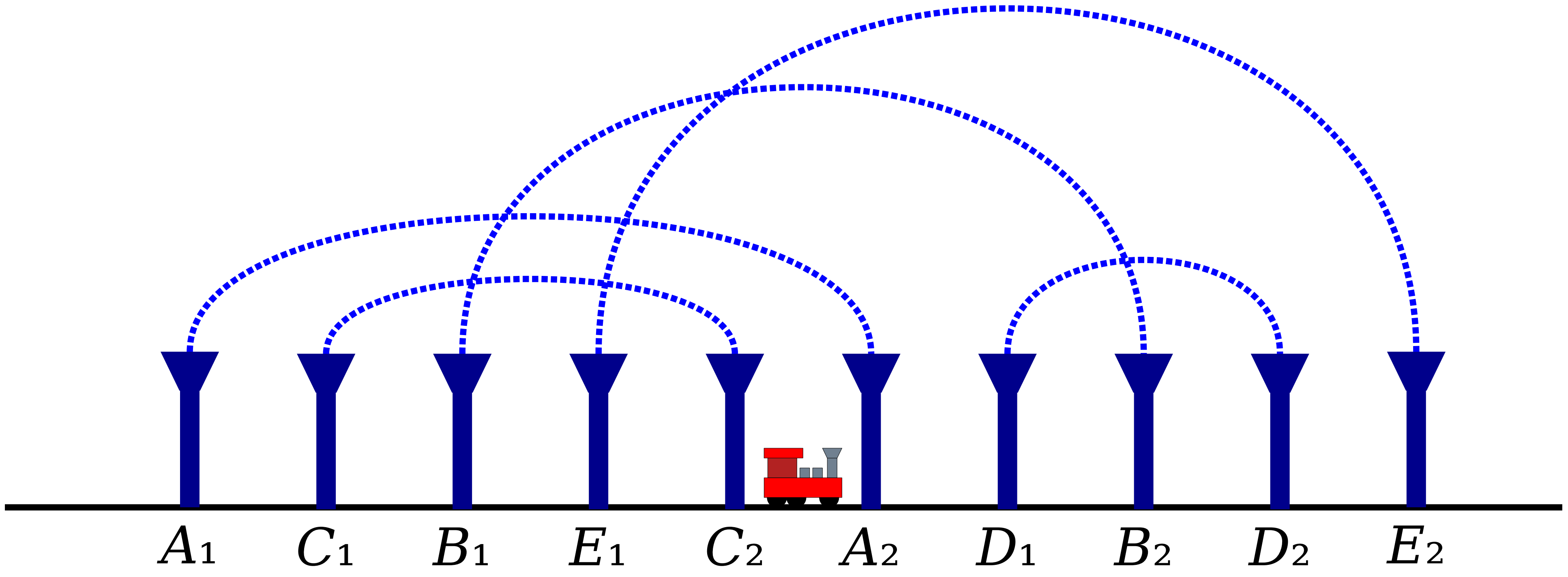

A_1
C_1
B_1
E_1
C_2
A_2
D_1
B_2
D_2
E_2

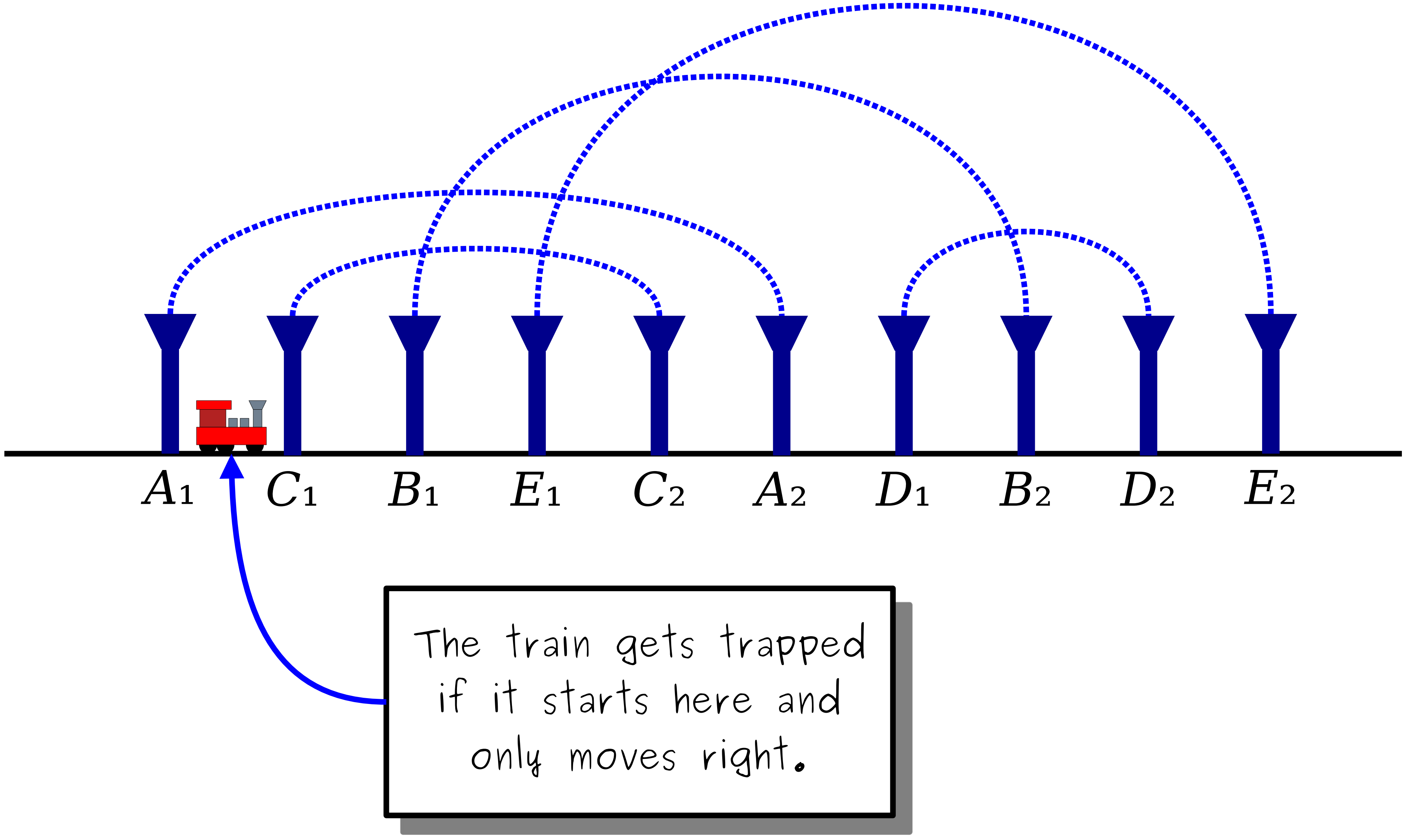

A_1
C_1
B_1
E_1
C_2
A_2
D_1
B_2
D_2
E_2


A_1
C_1
B_1
E_1
C_2
A_2
D_1
B_2
D_2
E_2


A_1
C_1
B_1
E_1
C_2
A_2
D_1
B_2
D_2
E_2

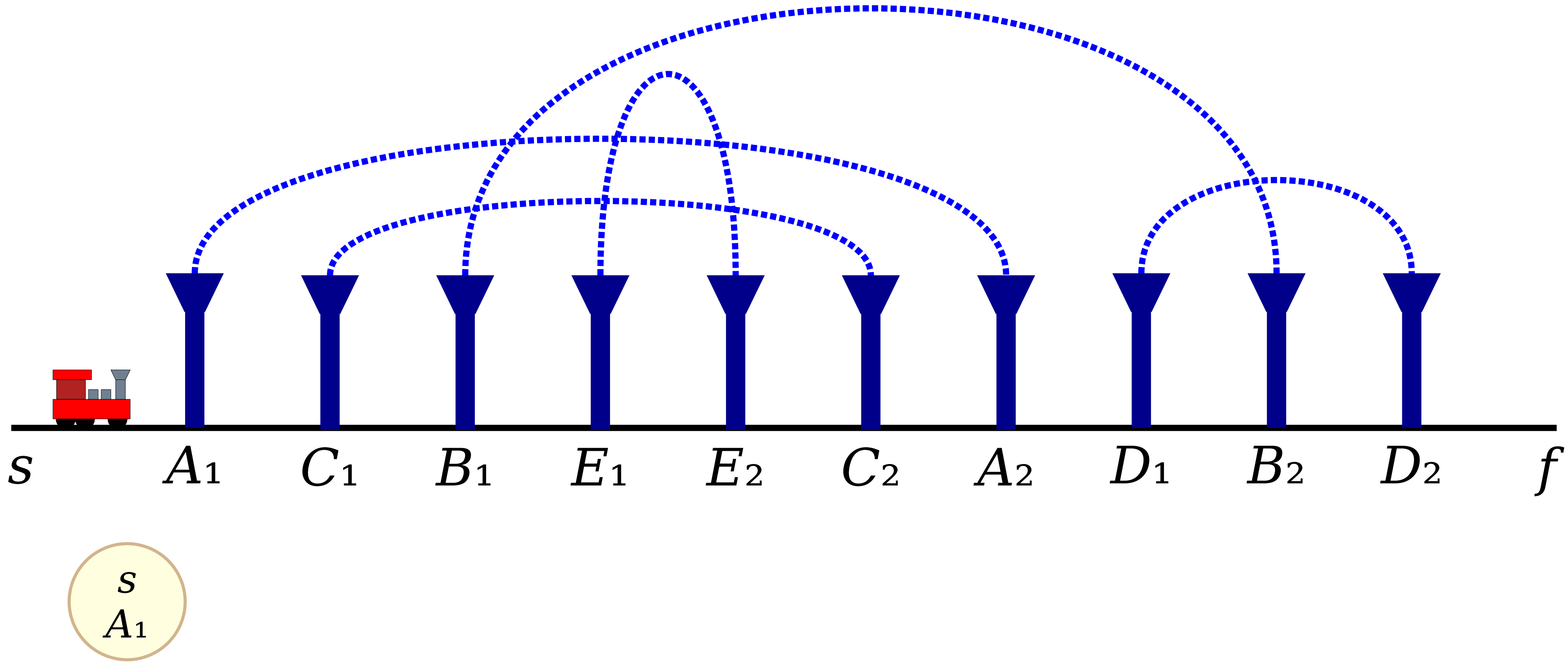

A_1
C_1
B_1
E_1
C_2
A_2
D_1
B_2
D_2
E_2


A_1
C_1
B_1
E_1
C_2
A_2
D_1
B_2
D_2
E_2

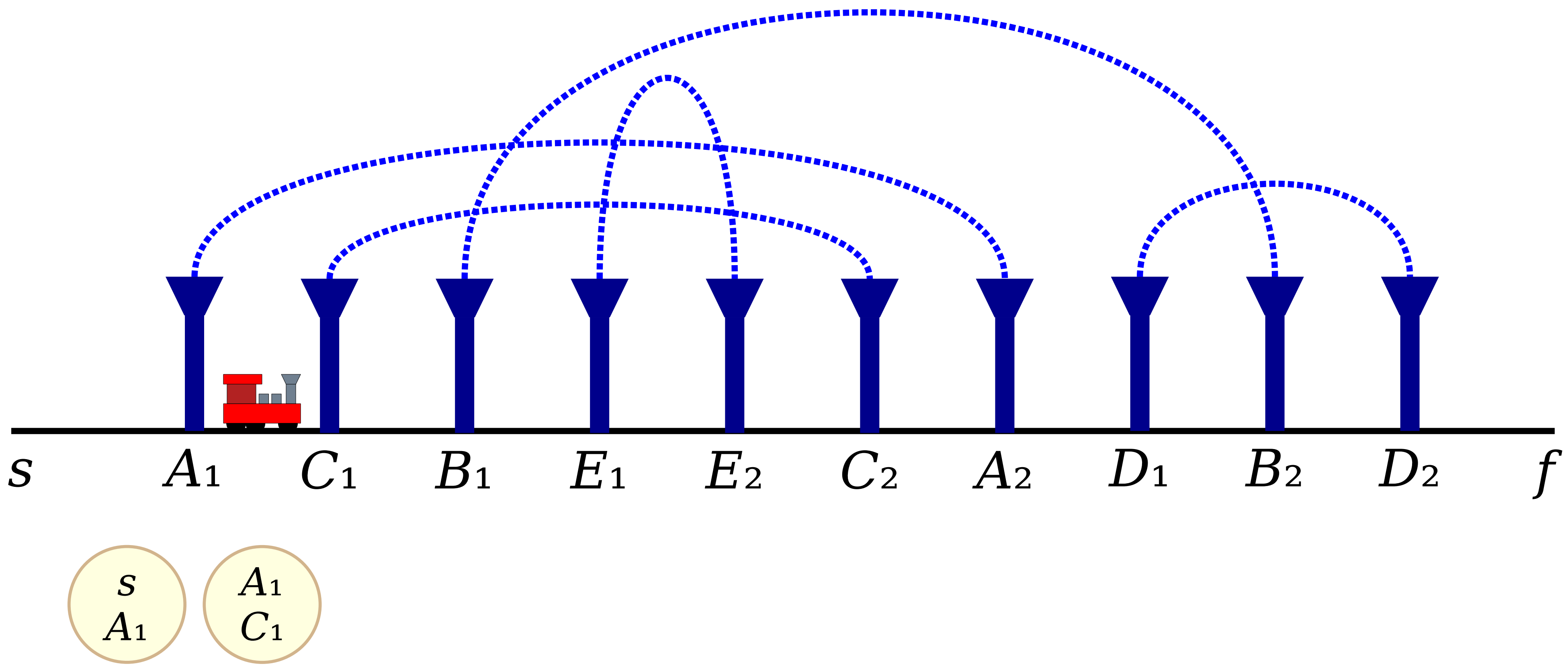

A_1
C_1
B_1
E_1
C_2
A_2
D_1
B_2
D_2
E_2
The train gets trapped if it starts here and only moves right.

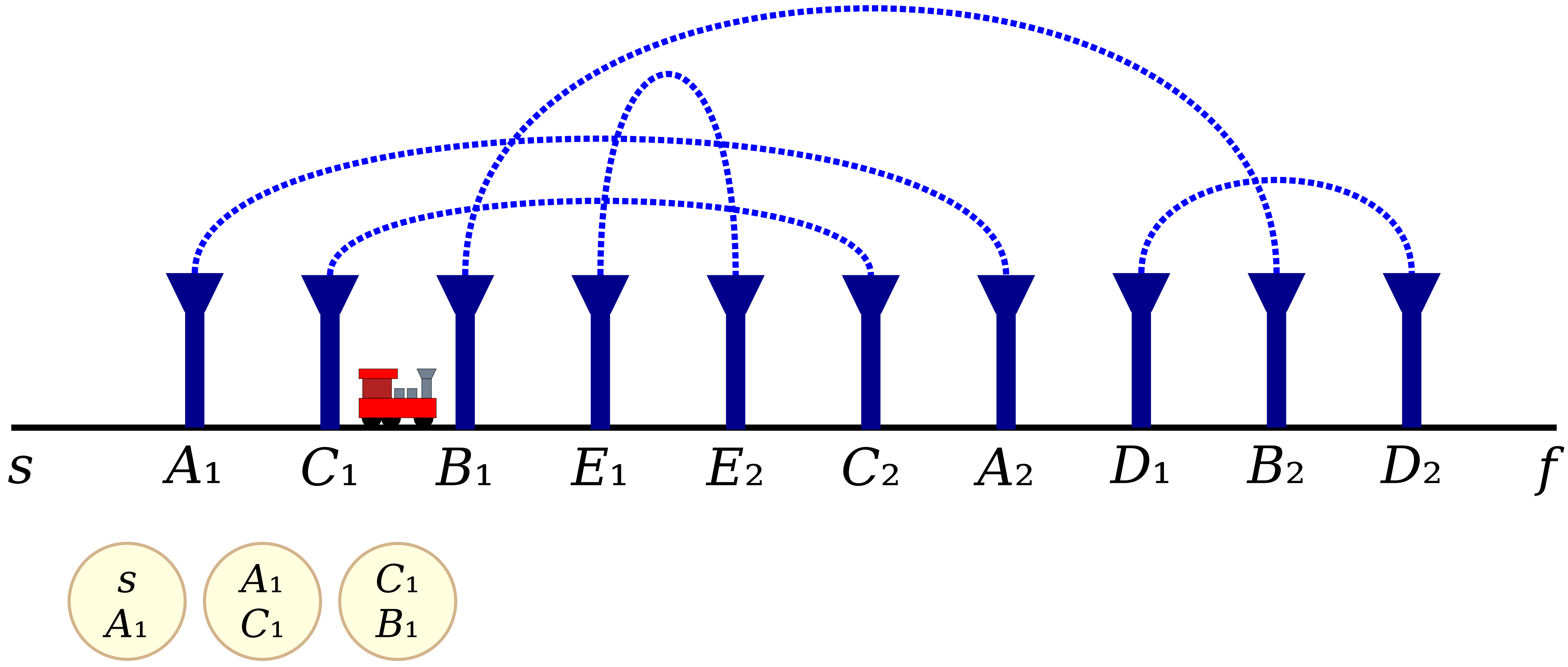
Can You Trap the Train?

- The train always drives to the right.
- The train starts just before the first teleporter.
- Teleporters always link in pairs.
- Teleporters can't stack on top of one another.
- Teleporters can't appear at or after the end point.
- You can use as many teleporters as you'd like.

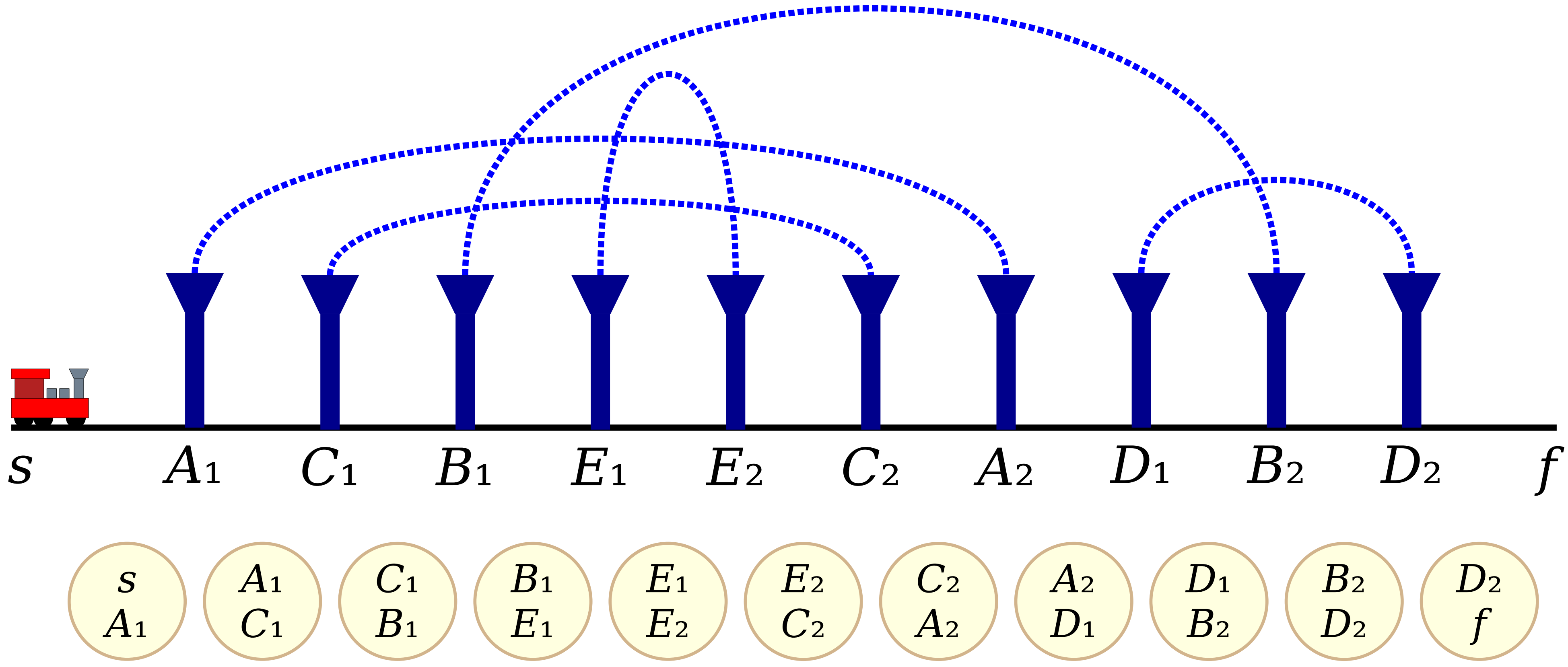


s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s
A_1

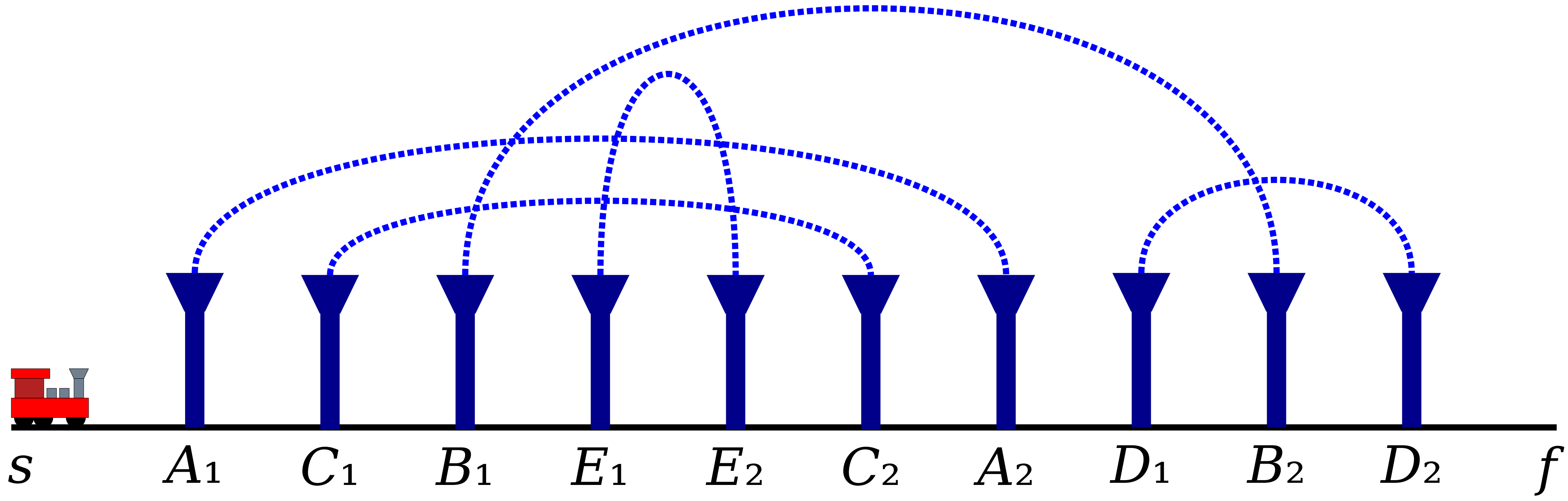

s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A1 C1



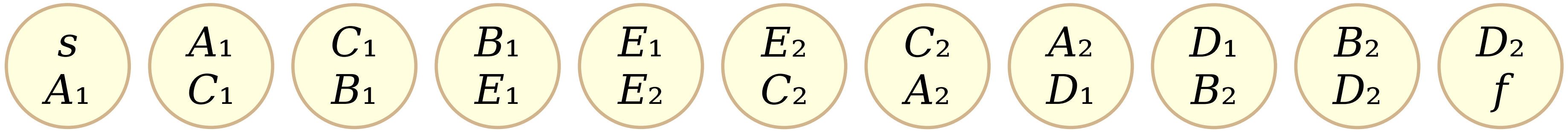
s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_1 C_1
C_1 B_1



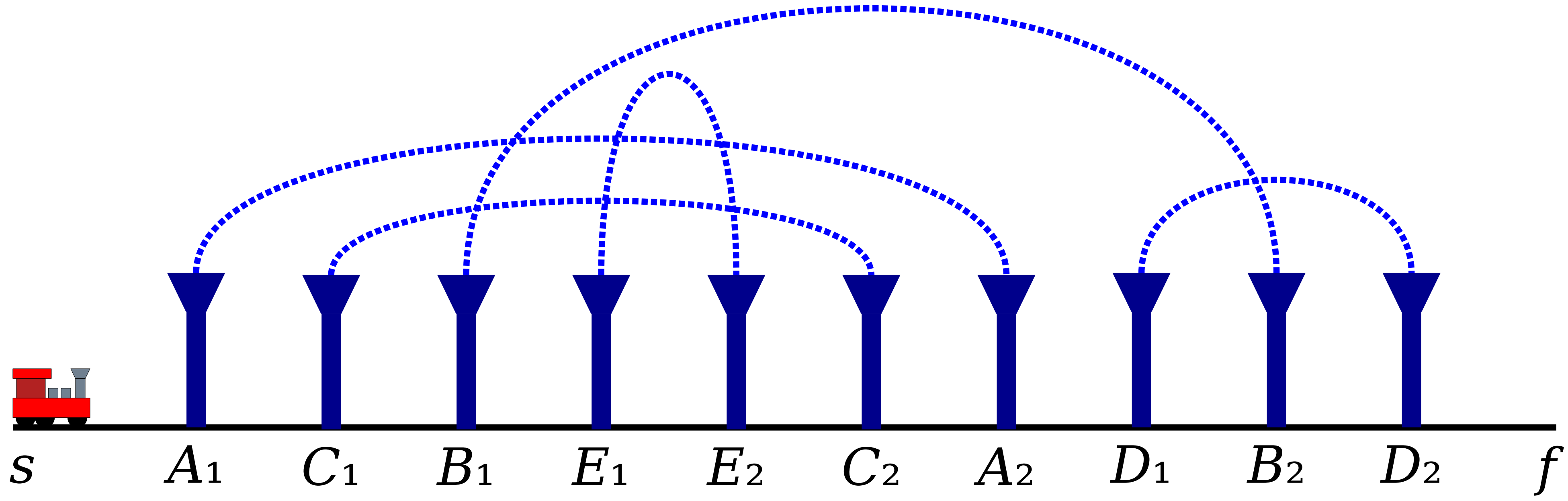
s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_1 C_1
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
C_2 A_2
A_2 D_1
D_1 B_2
B_2 D_2
D_2 f

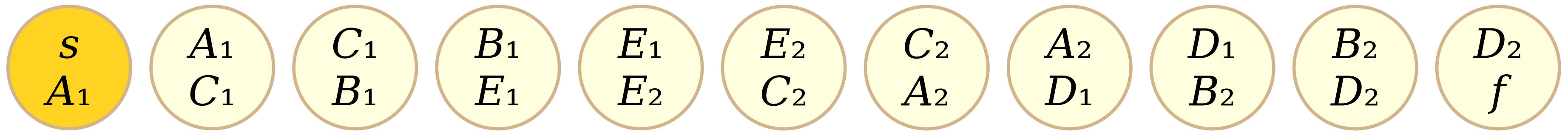


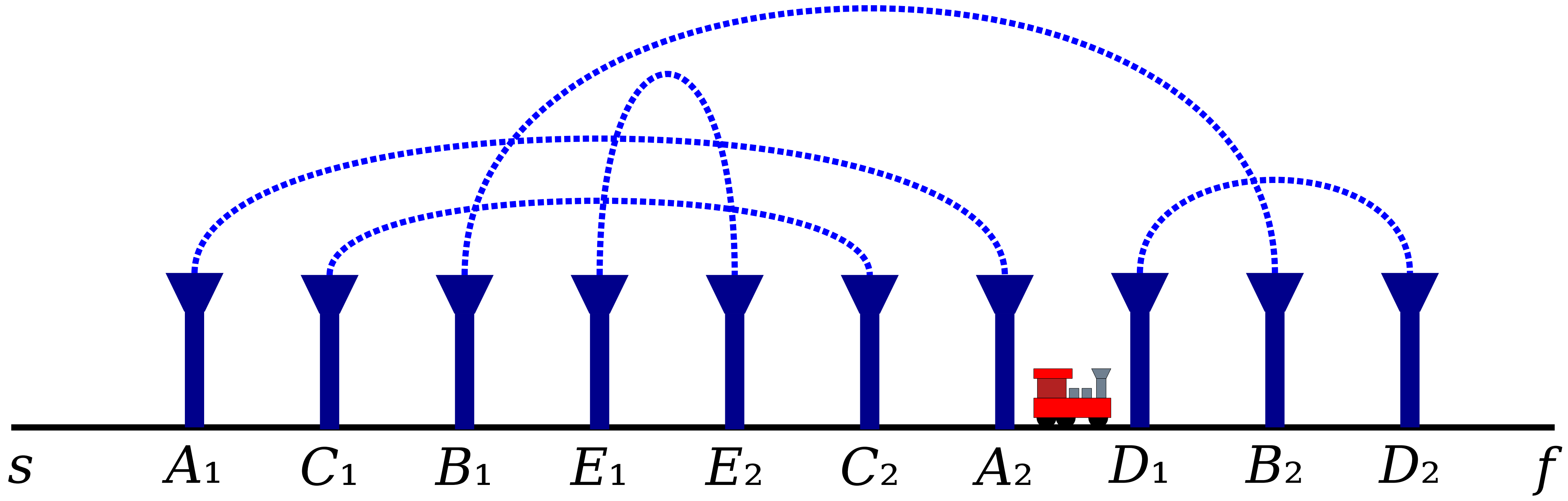
s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f

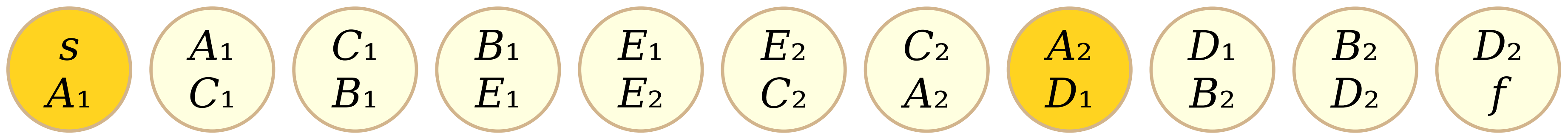


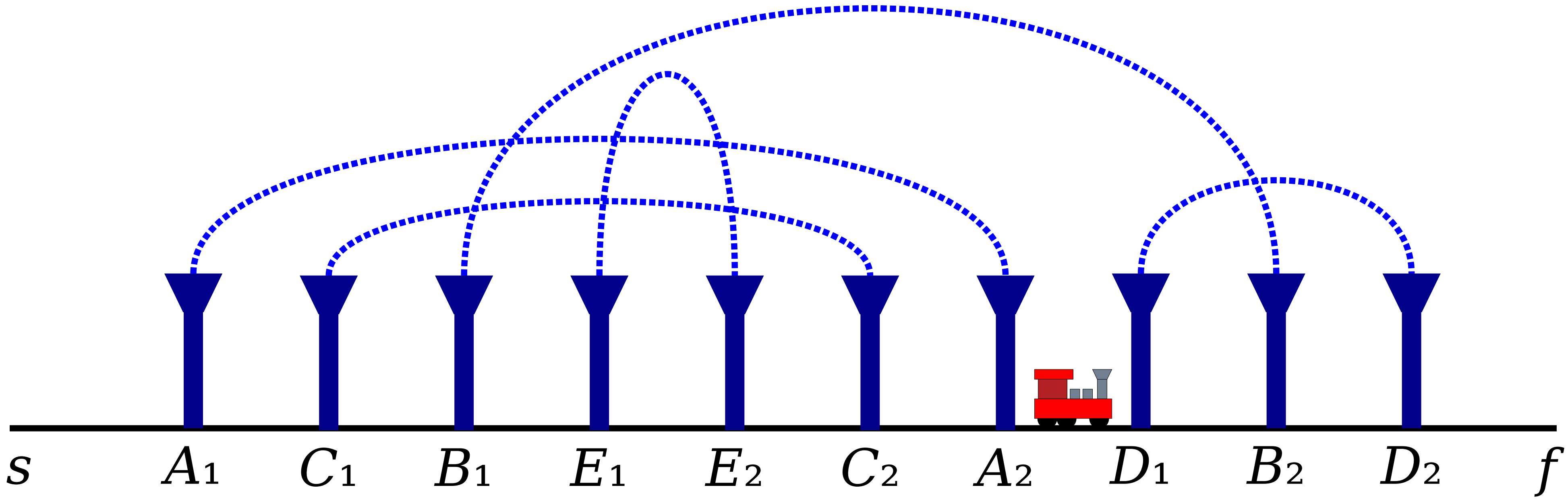
s A_1
A_1 C_1
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
C_2 A_2
A_2 D_1
D_1 B_2
B_2 D_2
D_2 f


s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f

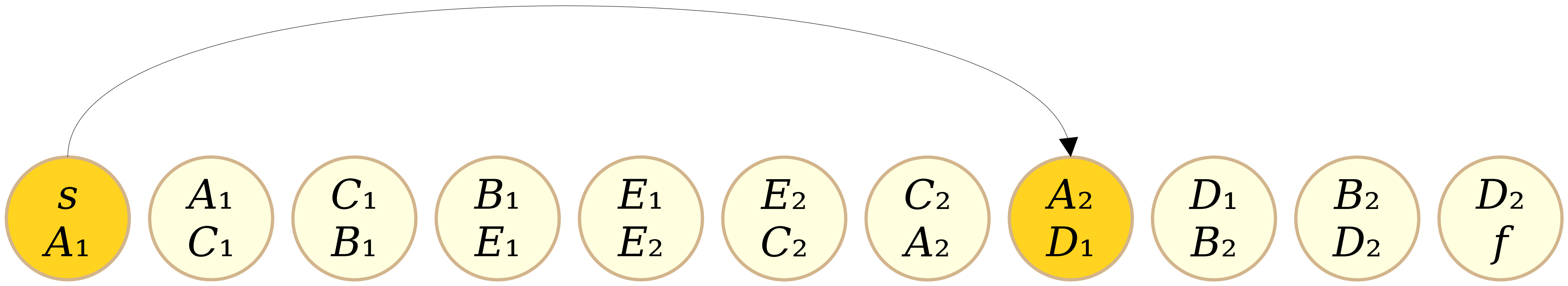

s A_1
A_1 C_1
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
C_2 A_2
A_2 D_1
D_1 B_2
B_2 D_2
D_2 f


s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f

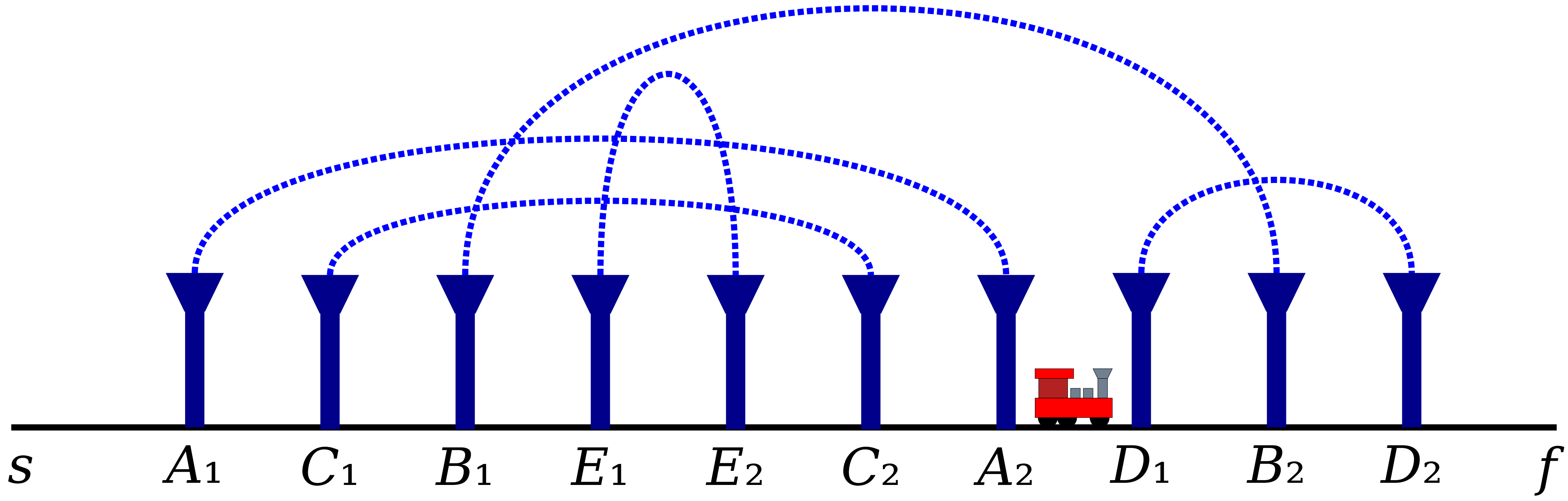

s A_1
A_1 C_1
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
C_2 A_2
A_2 D_1
D_1 B_2
B_2 D_2
D_2 f

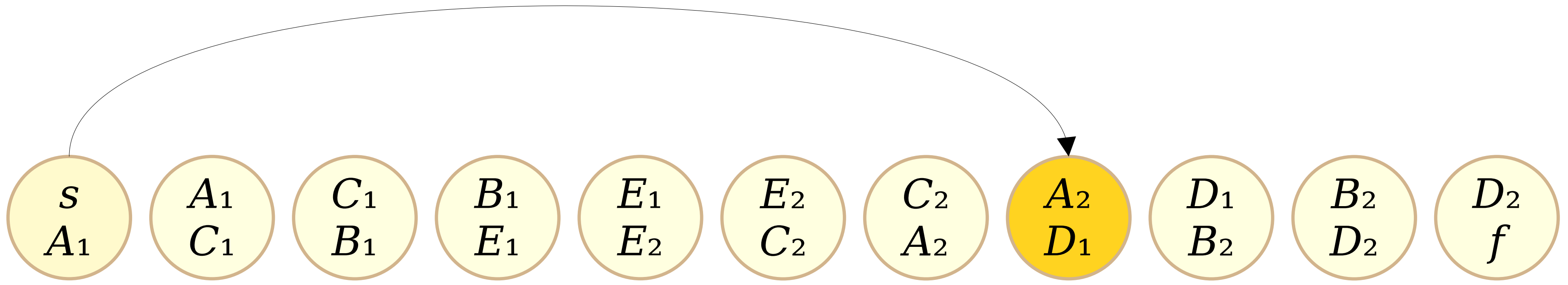


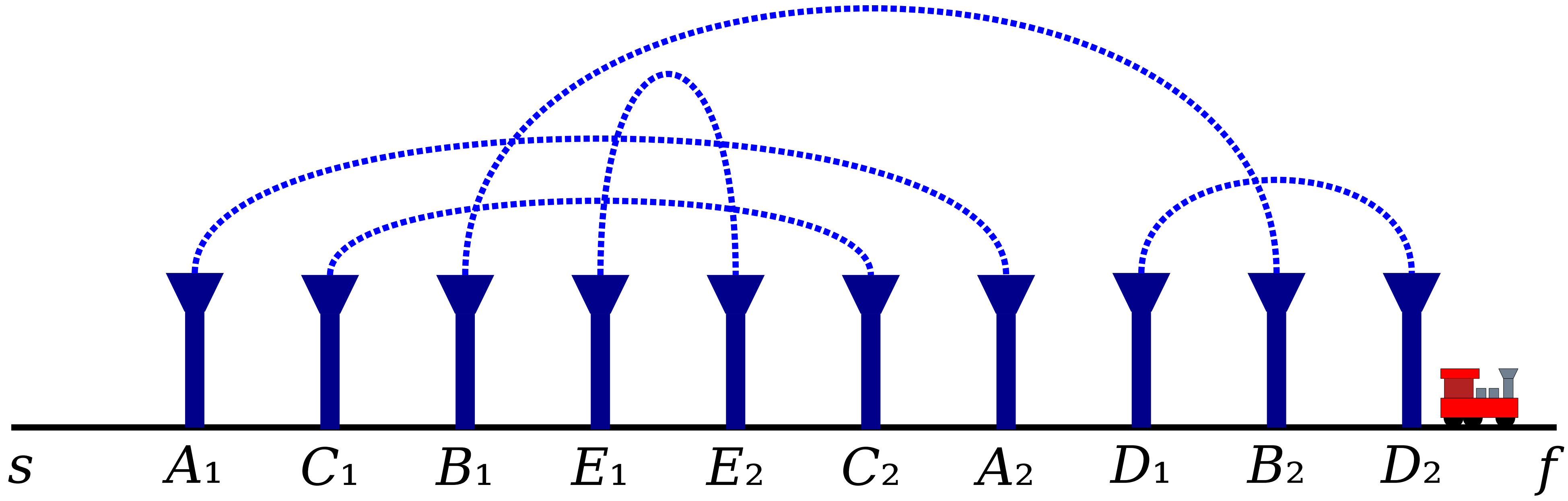
s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f



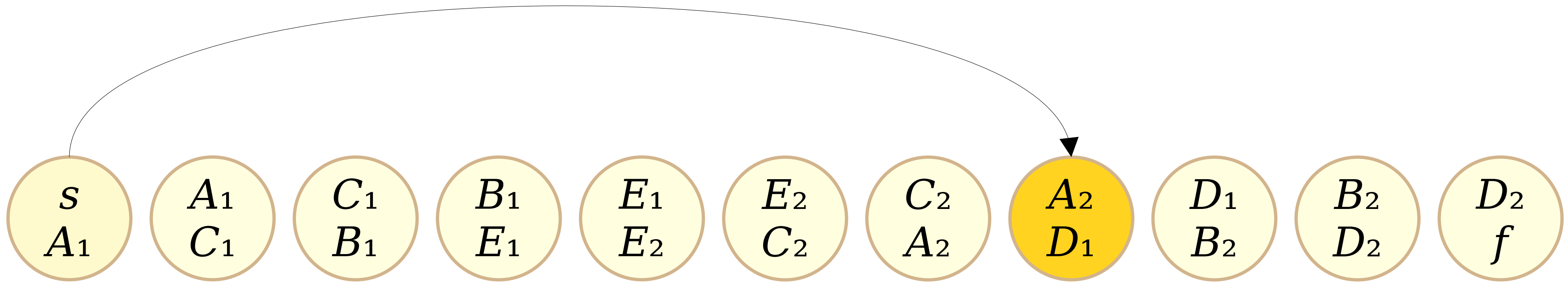
s A_1
A_1 C_1
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
C_2 A_2
A_2 D_1
D_1 B_2
B_2 D_2
D_2 f


s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f

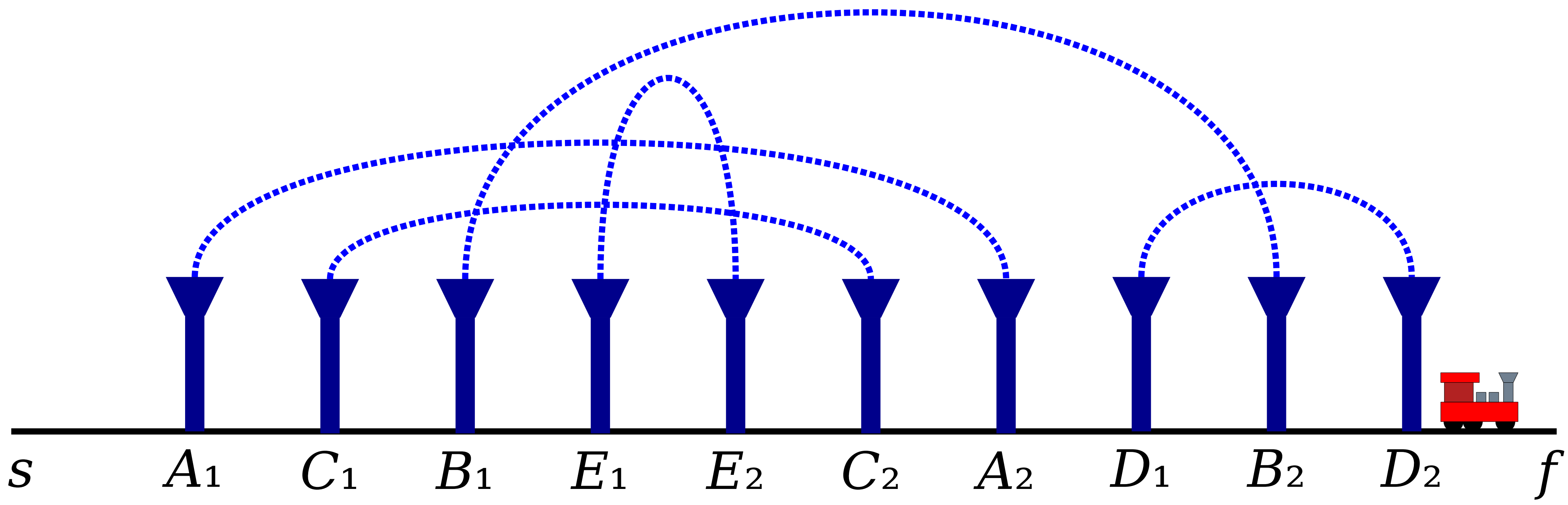

s A_1
A_1 C_1
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
C_2 A_2
A_2 D_1
D_1 B_2
B_2 D_2
D_2 f

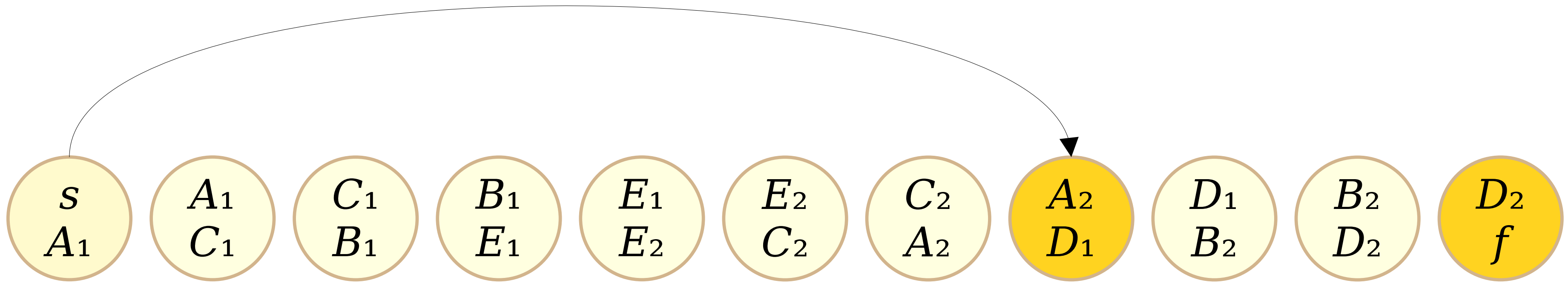


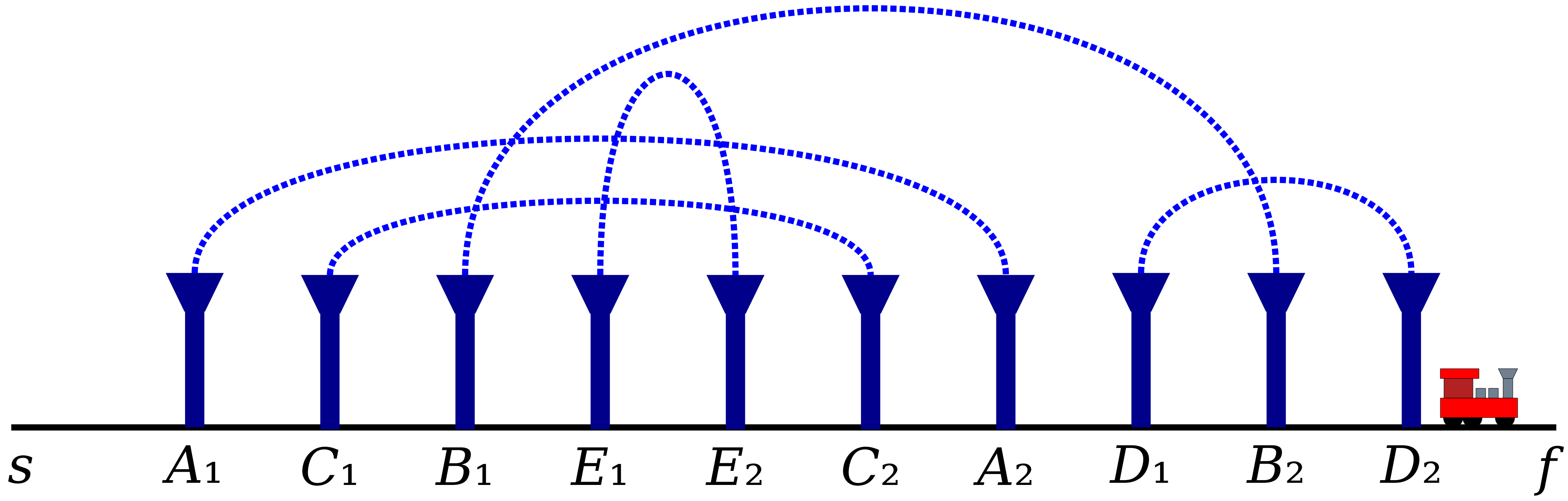
s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f

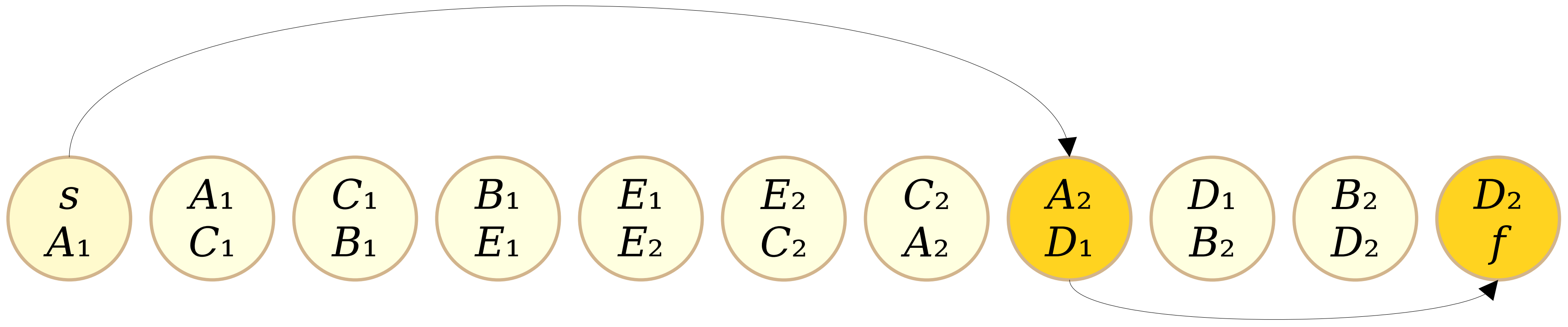


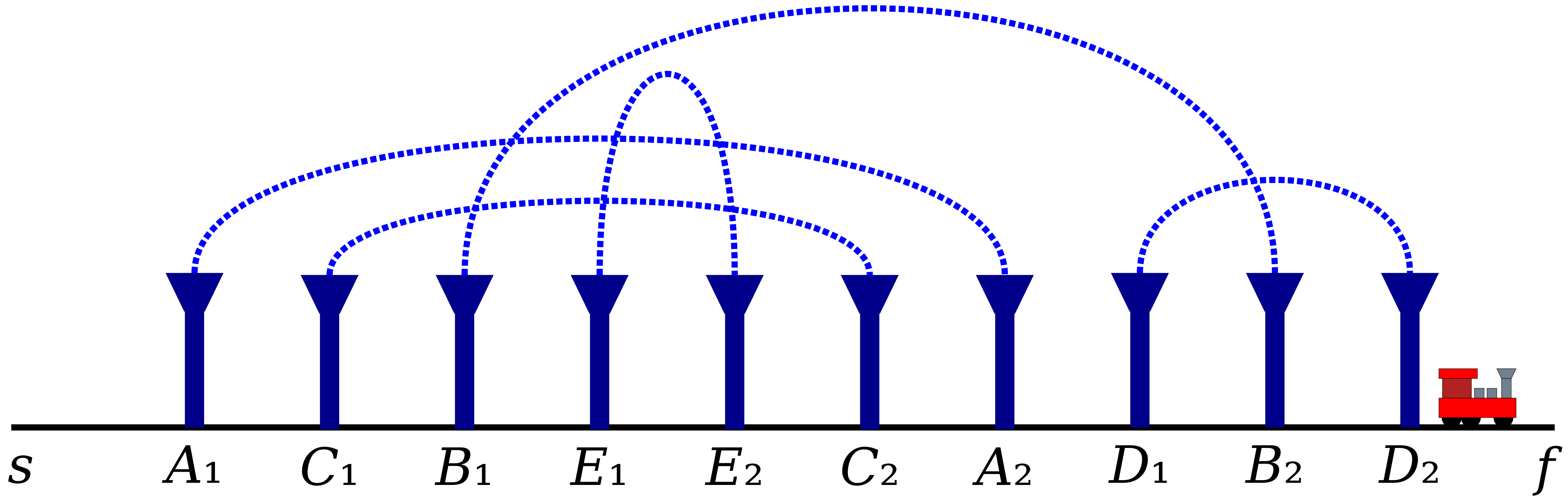
s A1
A1 C1
C1 B1
B1 E1
E1 E2
E2 C2
C2 A2
A2 D1
D1 B2
B2 D2
D2 f


s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f

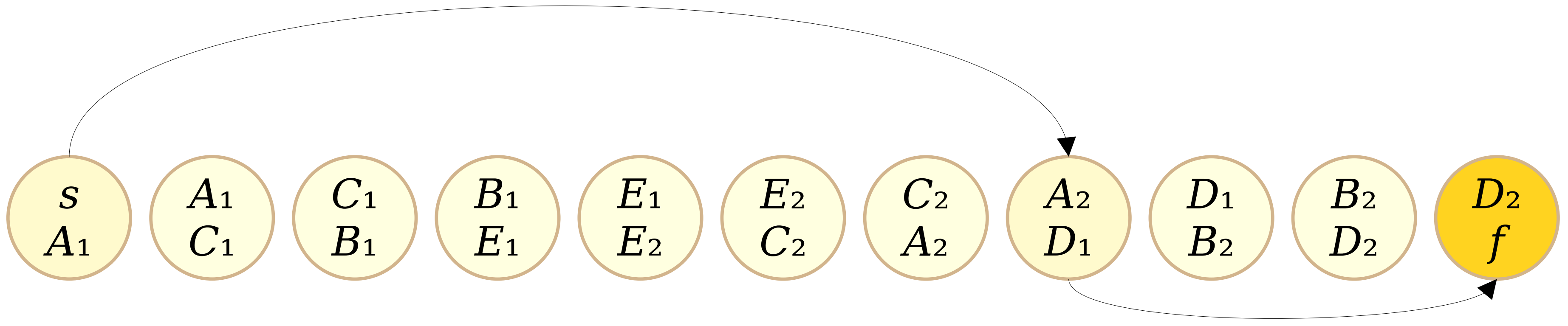

s A1
A1 C1
C1 B1
B1 E1
E1 E2
E2 C2
C2 A2
A2 D1
D1 B2
B2 D2
D2 f


s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f

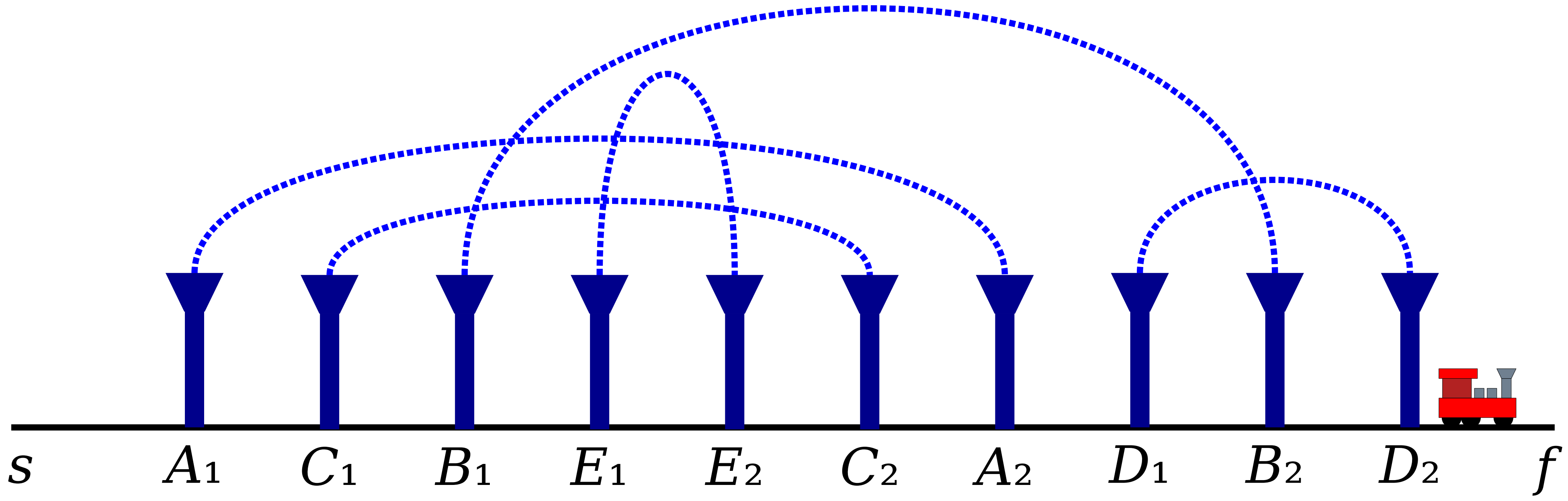

s A_1
A_1 C_1
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
C_2 A_2
A_2 D_1
D_1 B_2
B_2 D_2
D_2 f

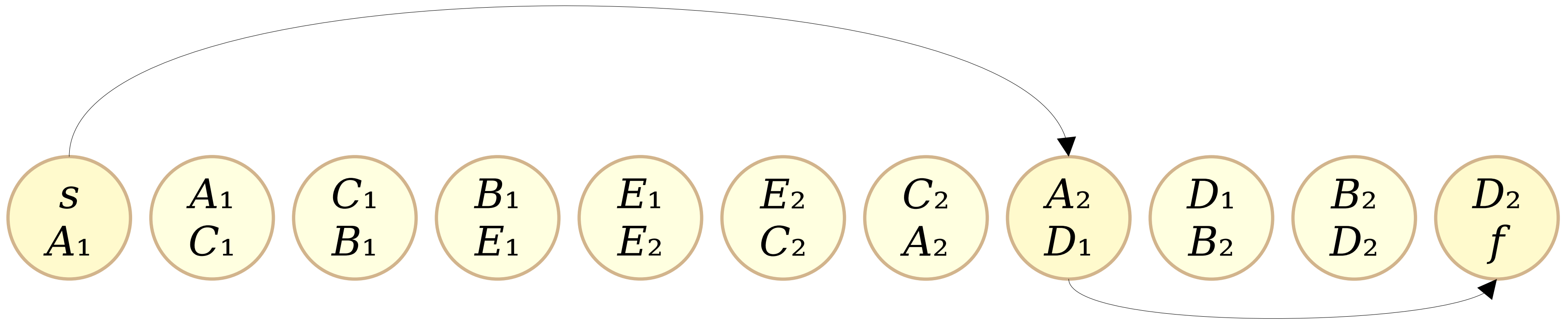


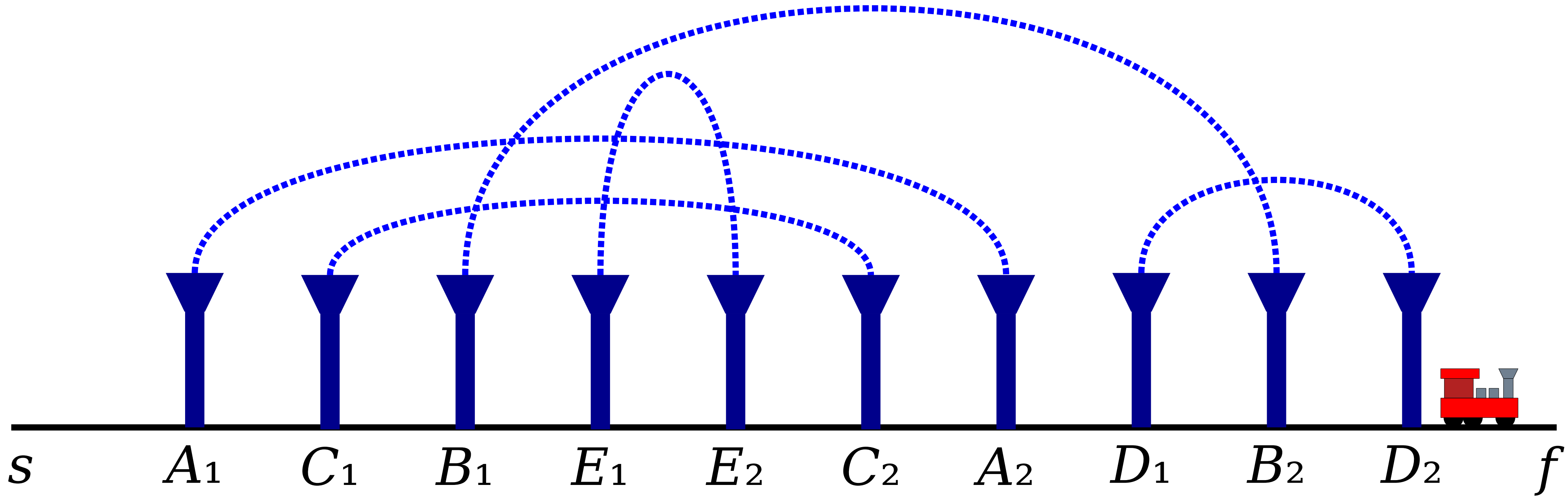
s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f



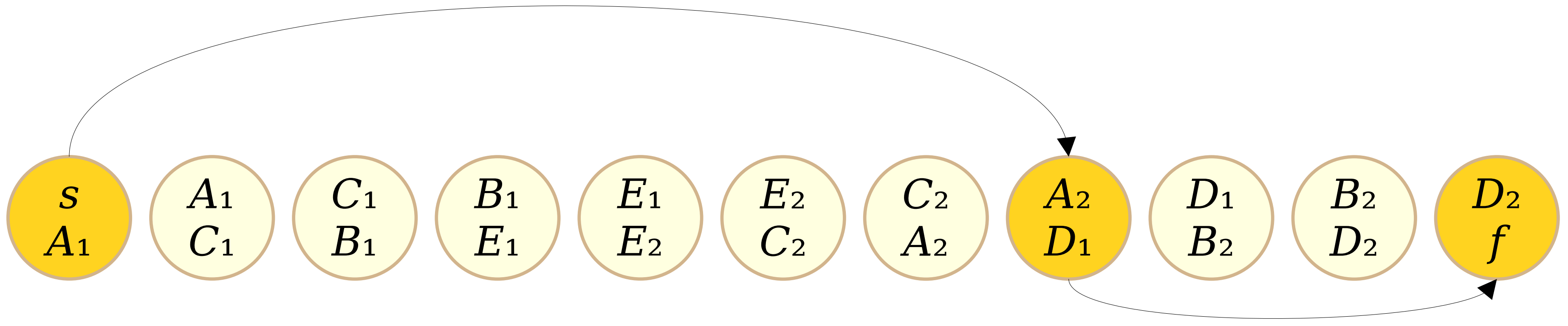
s A_1
A_1 C_1
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
C_2 A_2
A_2 D_1
D_1 B_2
B_2 D_2
D_2 f


s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f

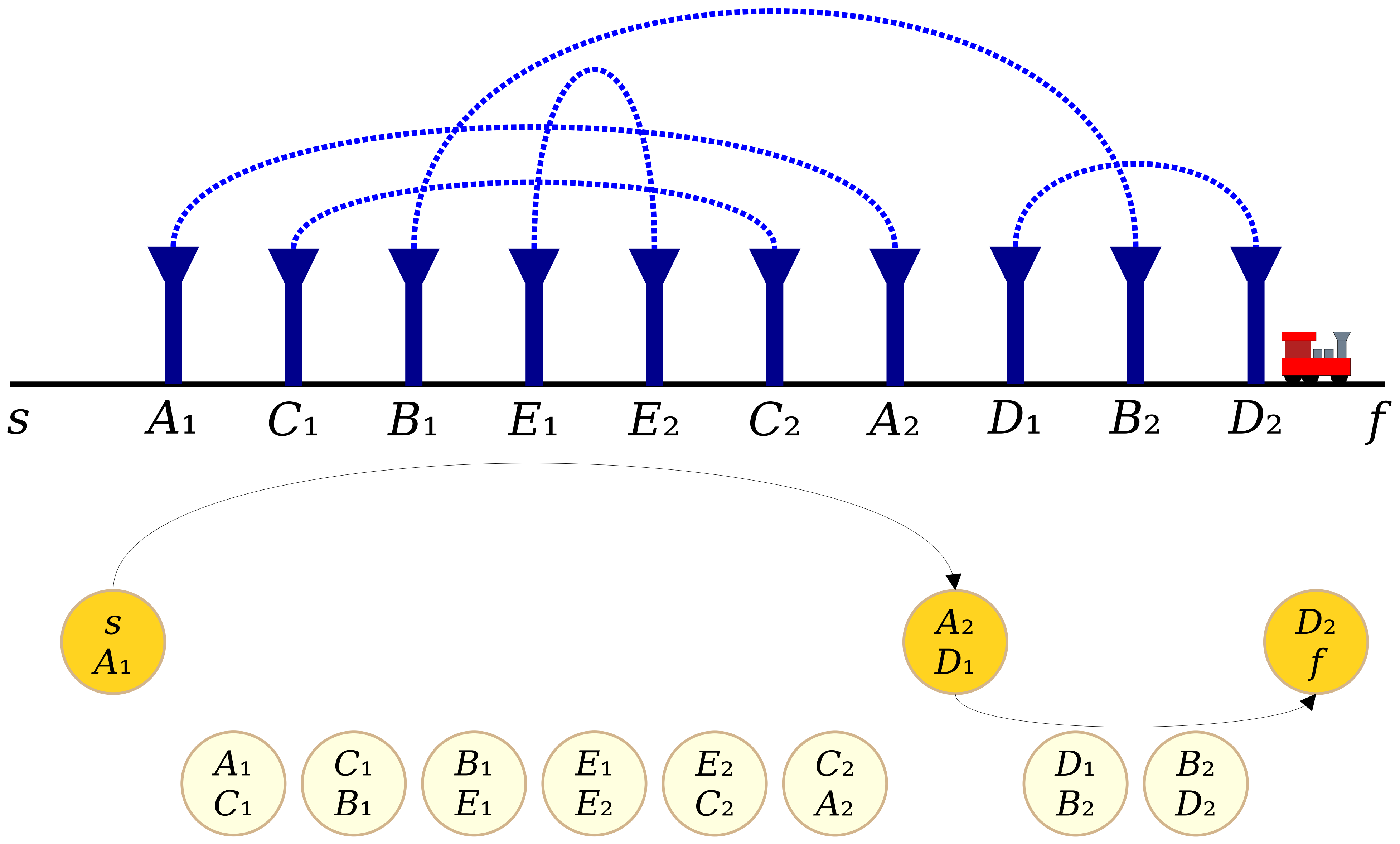

s A_1
A_1 C_1
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
C_2 A_2
A_2 D_1
D_1 B_2
B_2 D_2
D_2 f

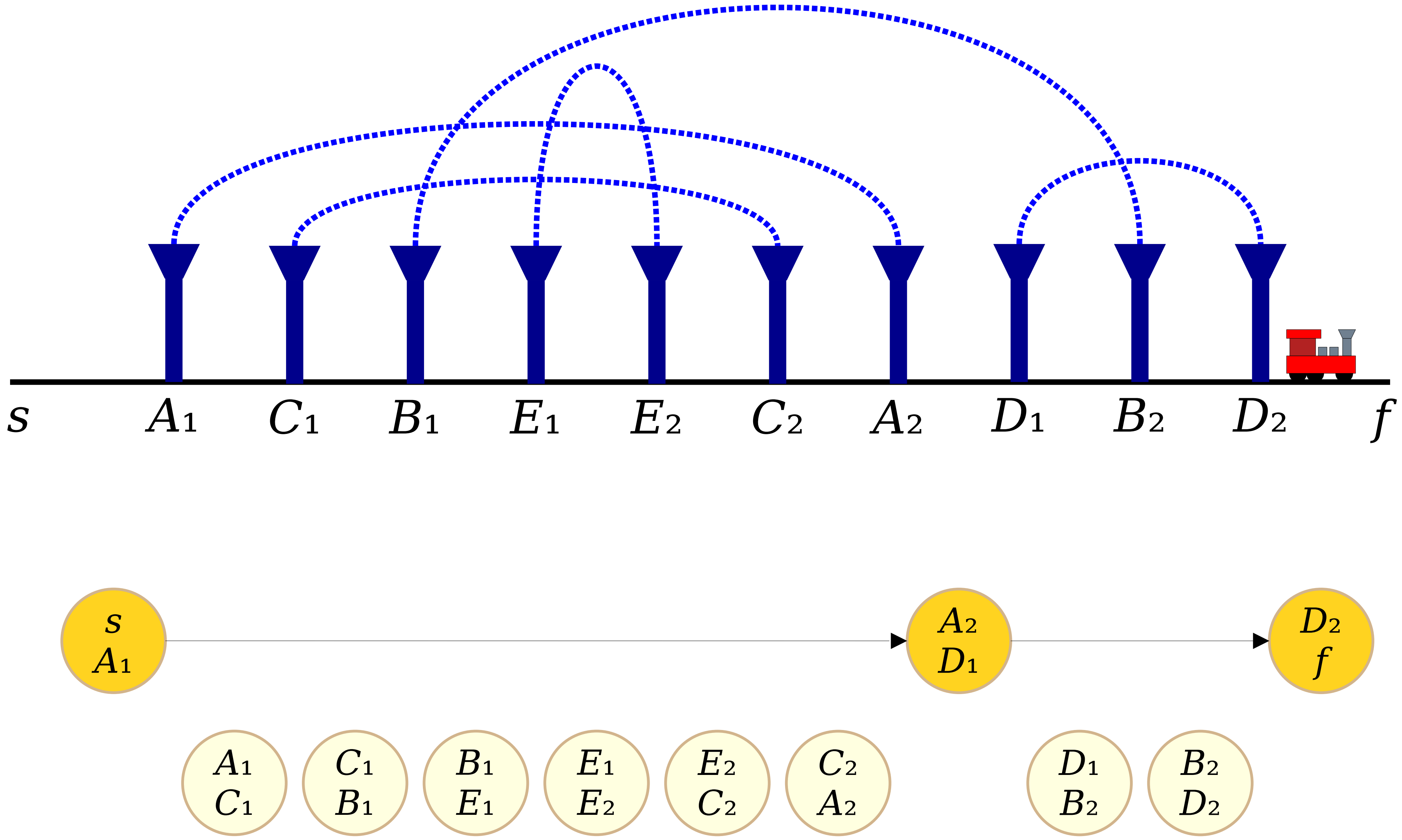


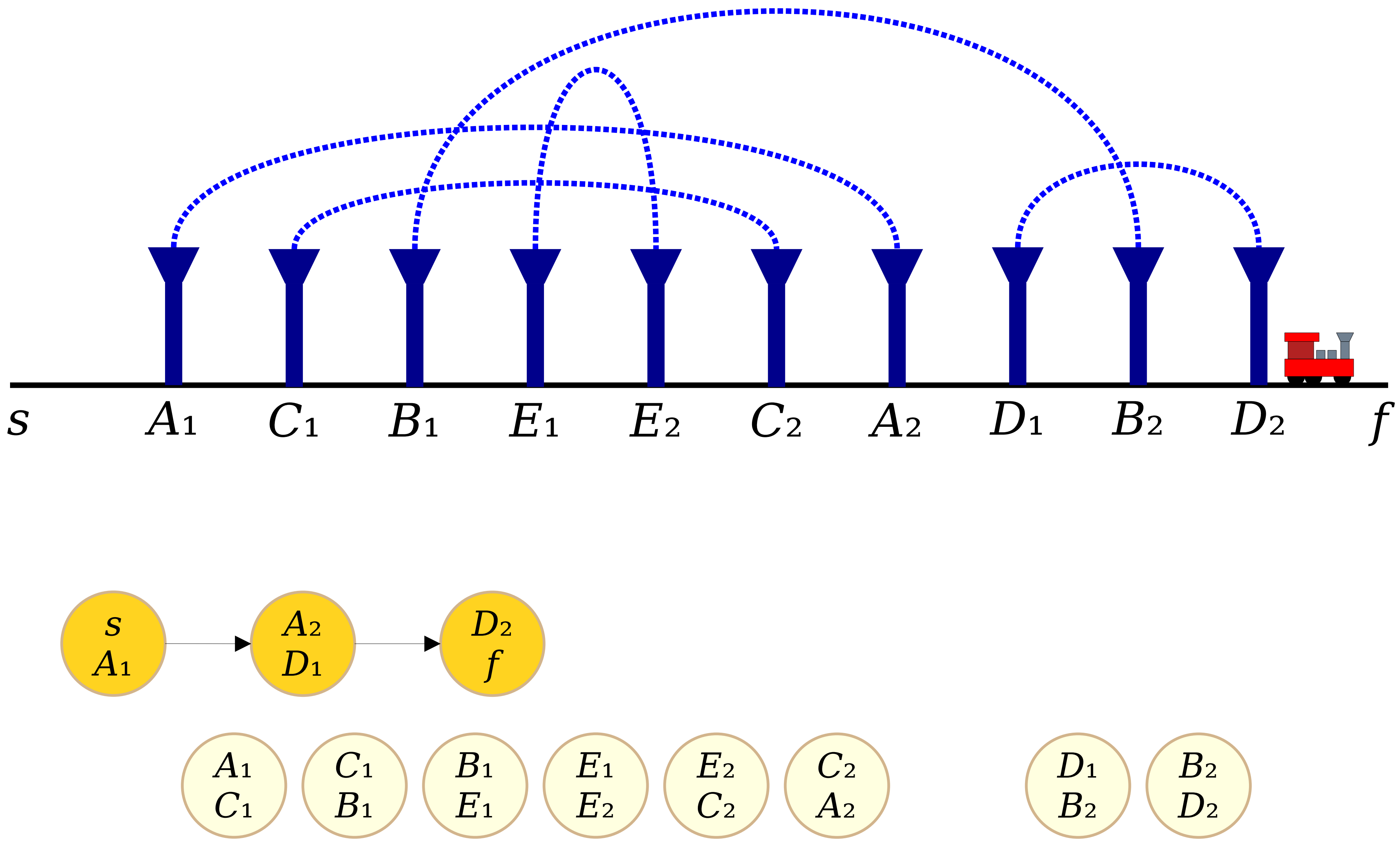
s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f



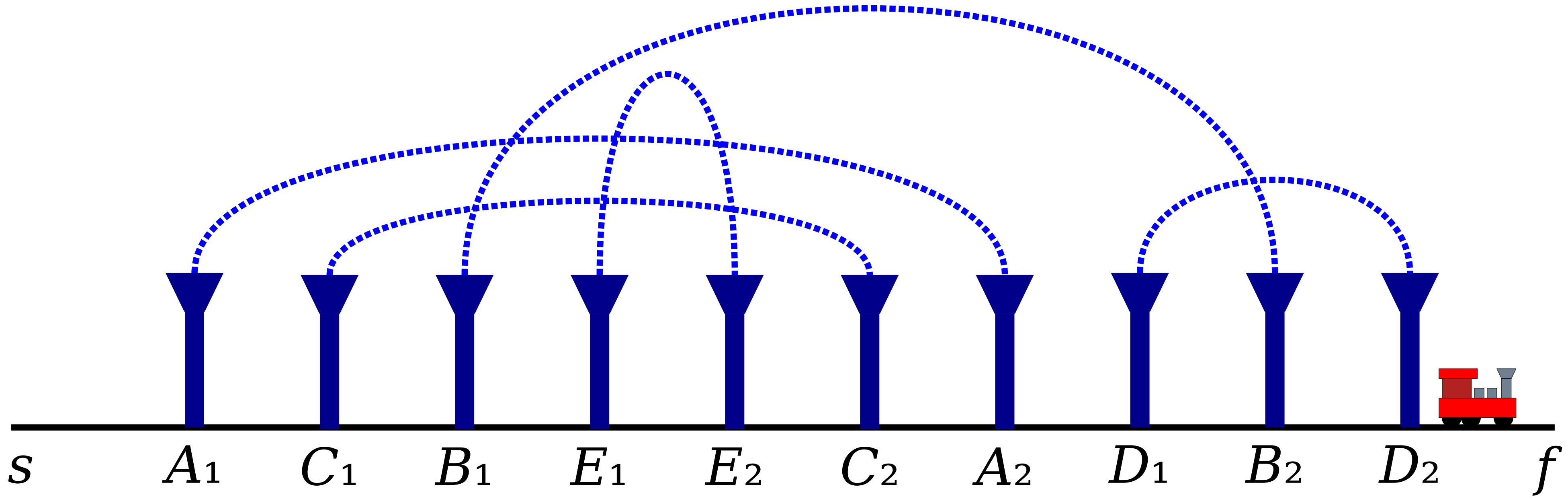
s A1
A1 C1
C1 B1
B1 E1
E1 E2
E2 C2
C2 A2
A2 D1
D1 B2
B2 D2
D2 f


s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C1 B1
B1 E1
E1 E2
E2 C2
C2 A2
D1 B2
B2 D2

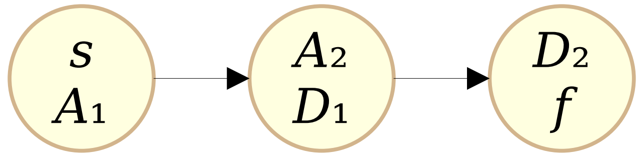

s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C1 B1
B1 E1
E1 E2
E2 C2
C2 A2
D1 B2
B2 D2



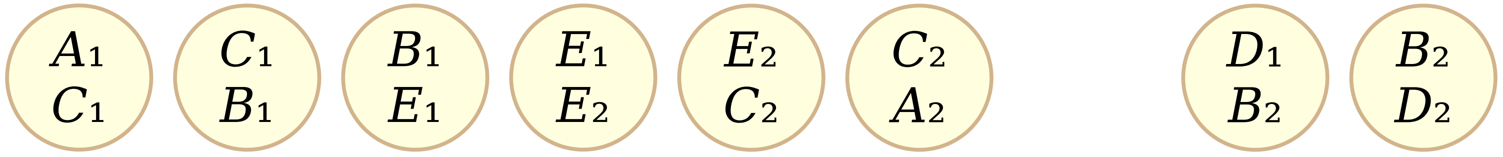
s
A₁
C₁
B₁
E₁
E₂
C₂
A₂
D₁
B₂
D₂
f
s A₁
A₂ D₁
D₂ f
A₁ C₁
C₁ B₁
B₁ E₁
E₁ E₂
E₂ C₂
C₂ A₂
D₁ B₂
B₂ D₂



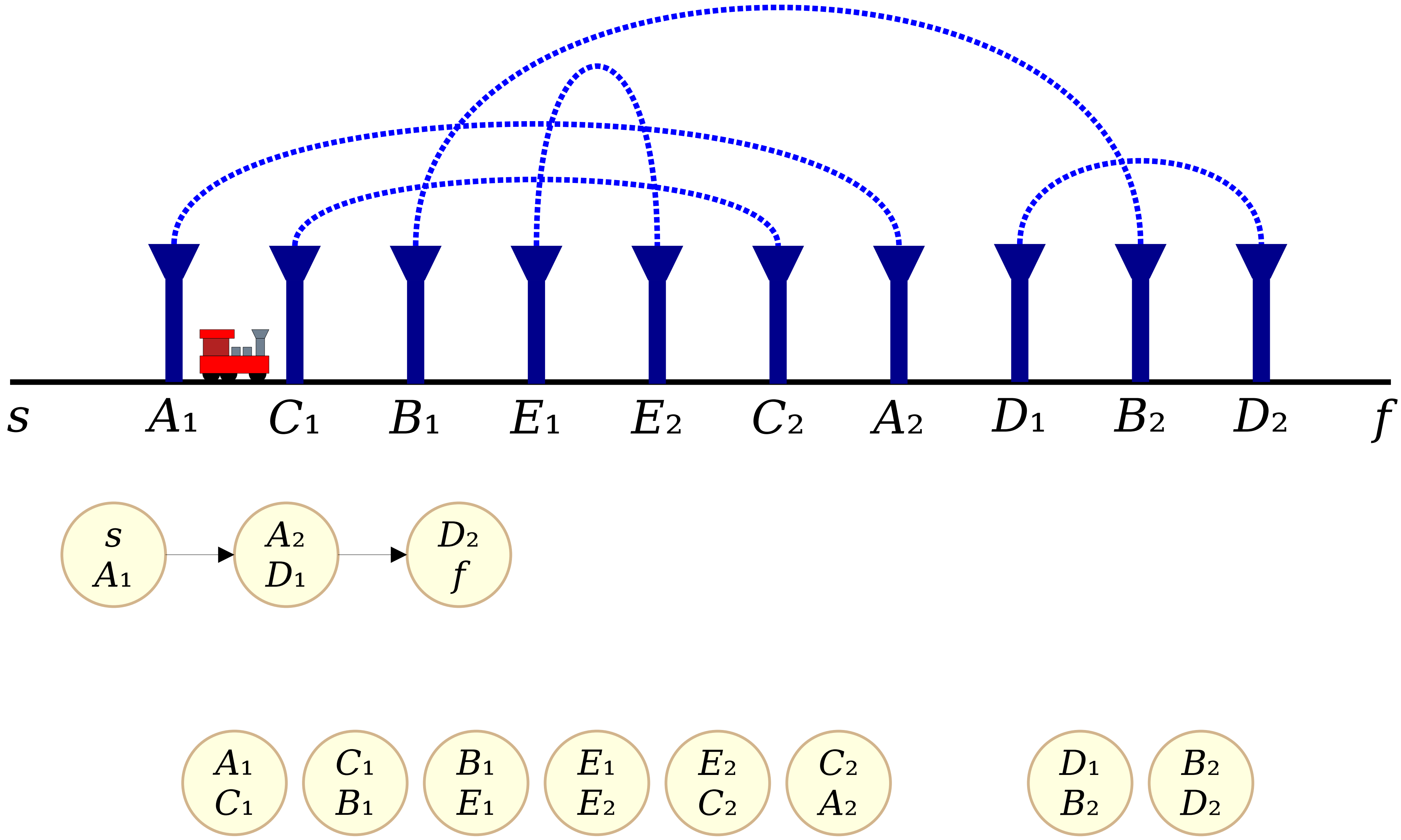
s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f

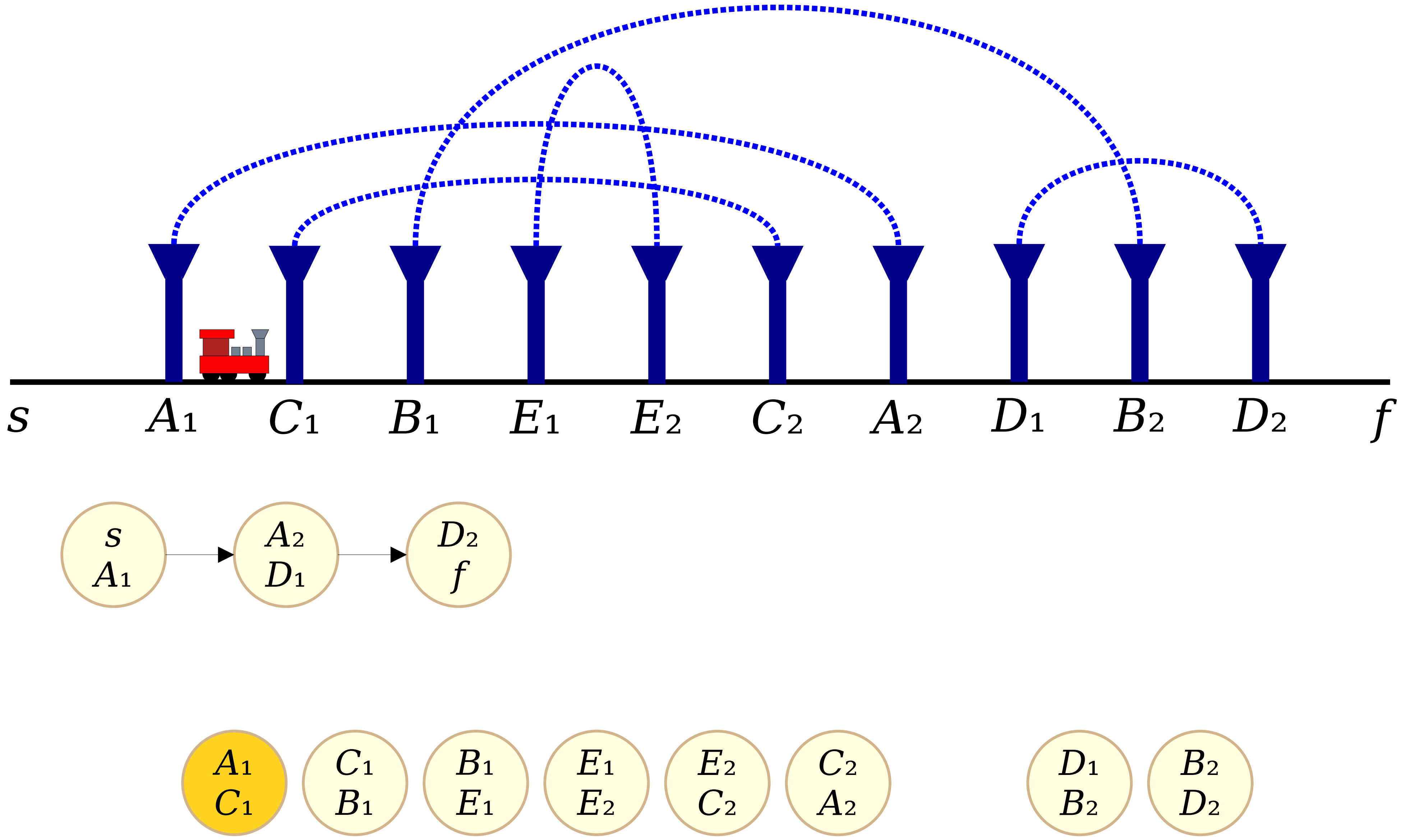


s A1
A2 D1
D2 f

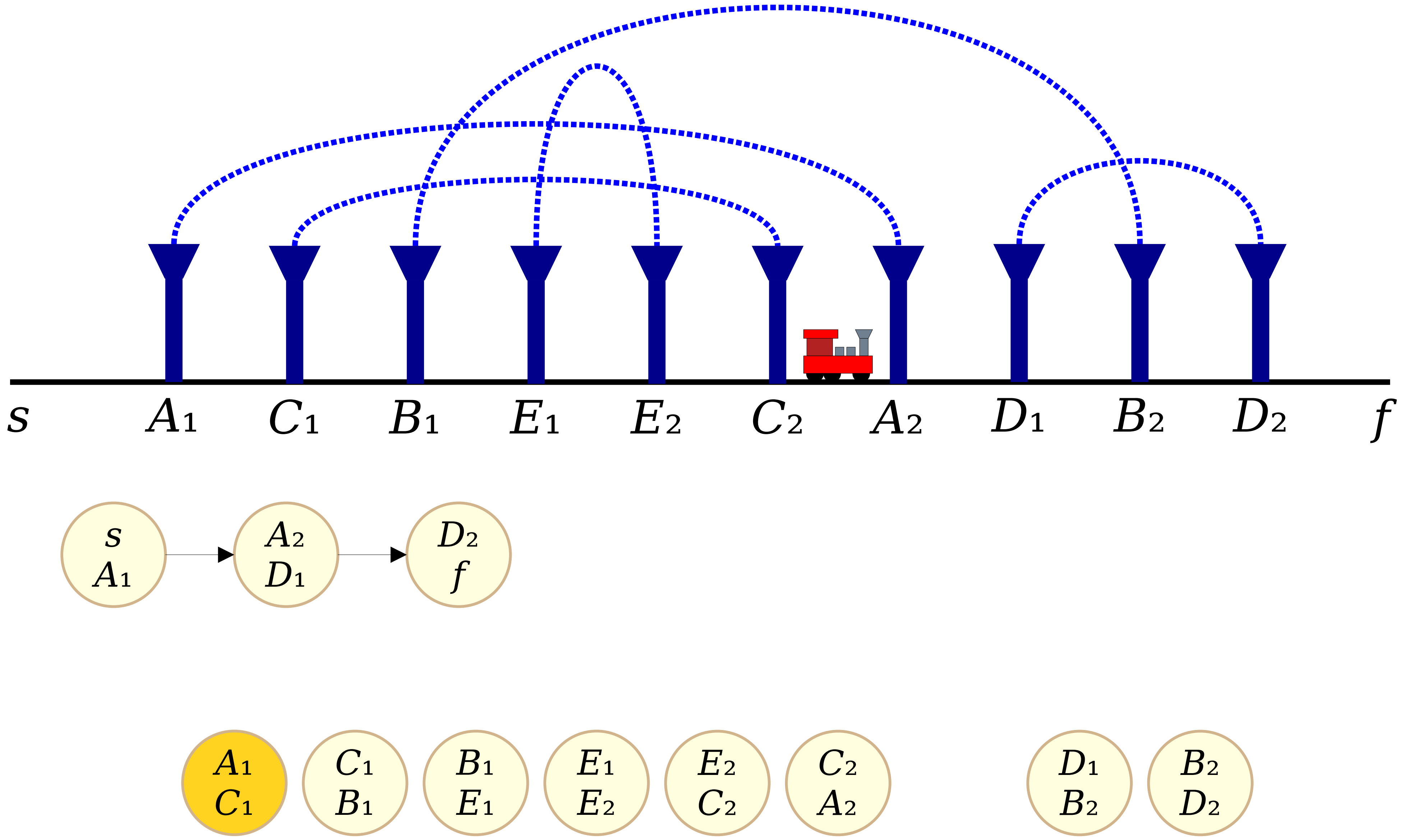


A1 C1
C1 B1
B1 E1
E1 E2
E2 C2
C2 A2
D1 B2
B2 D2

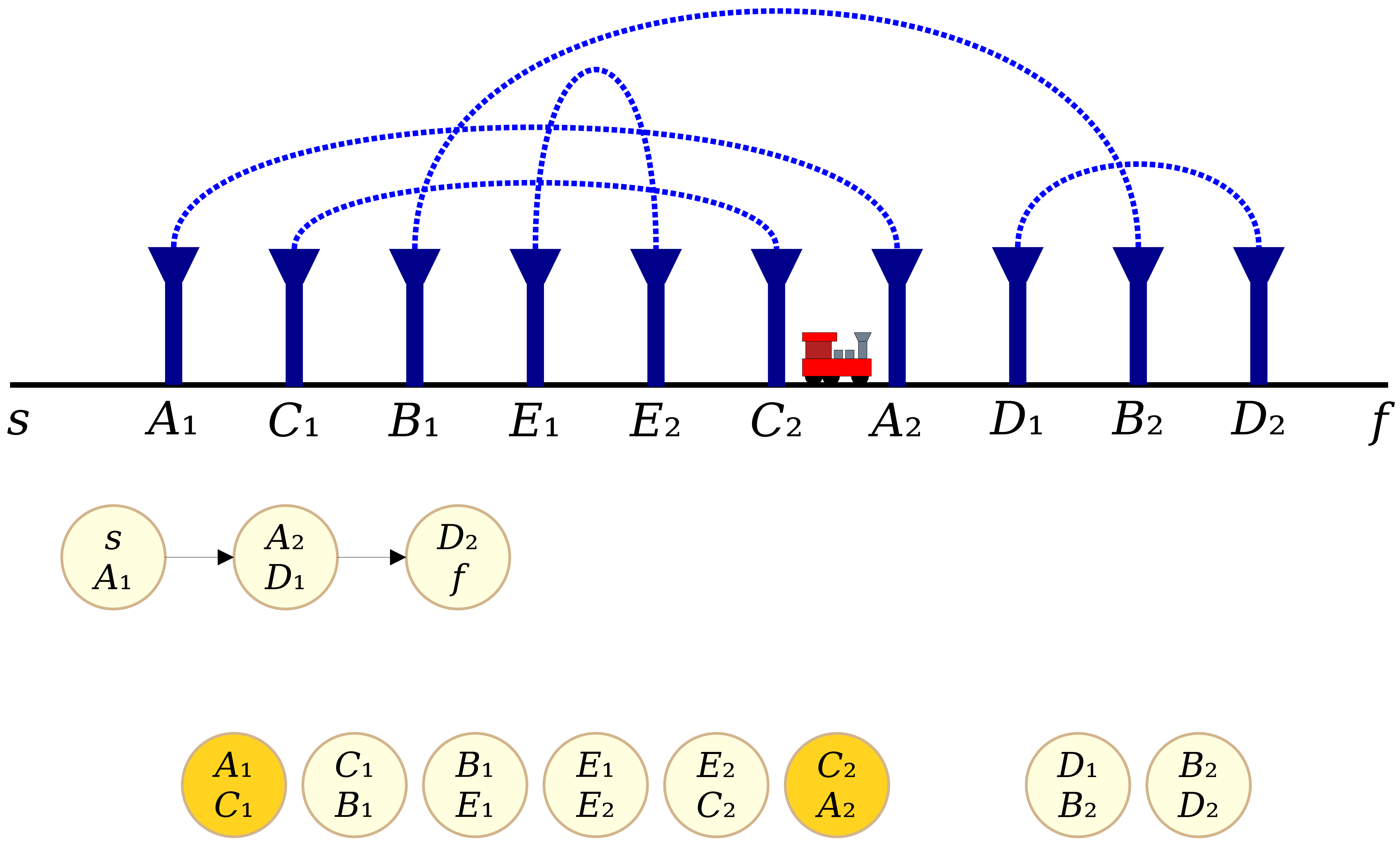

s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C1 B1
B1 E1
E1 E2
E2 C2
C2 A2
D1 B2
B2 D2



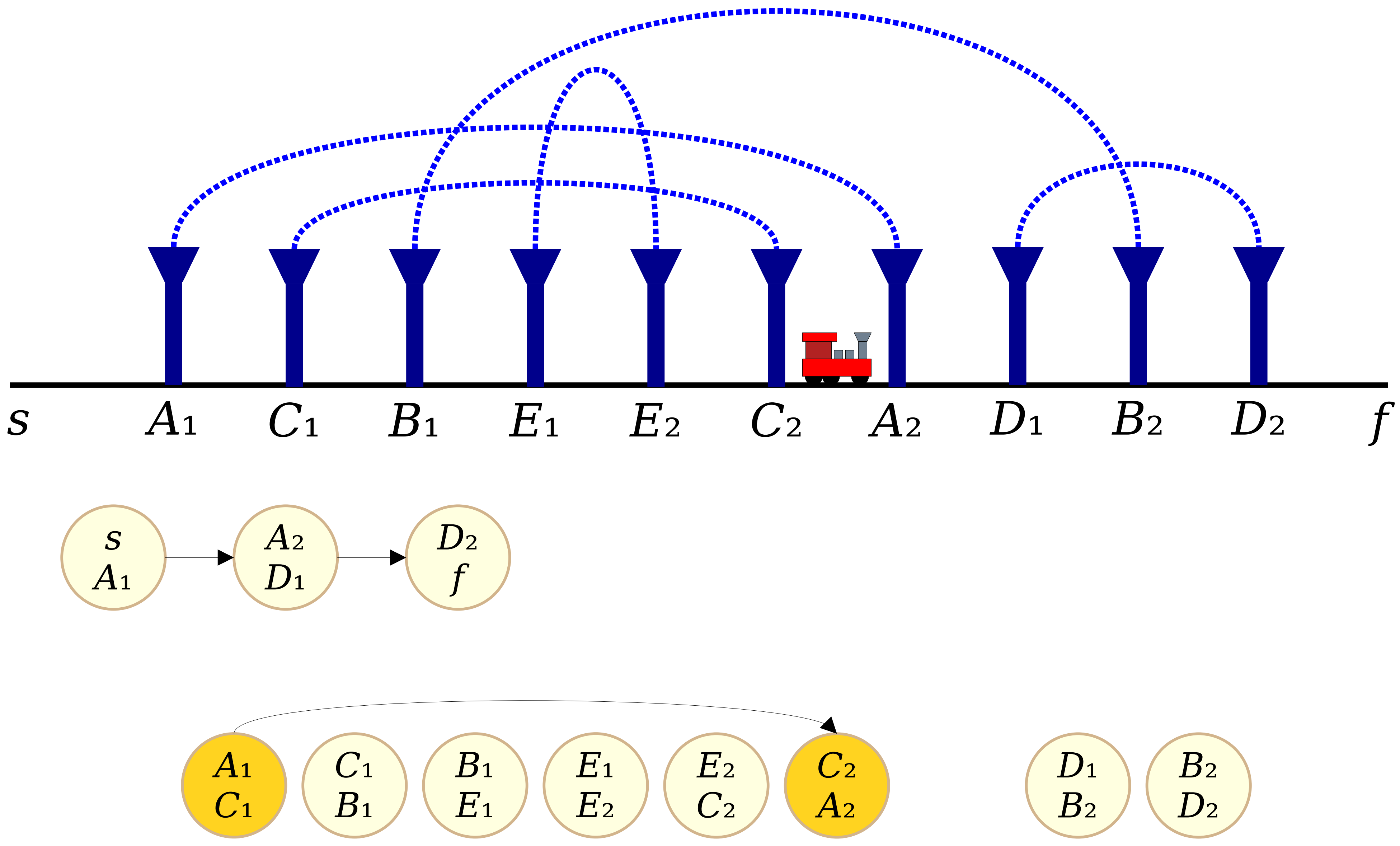
s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
C_2 A_2
D_1 B_2
B_2 D_2



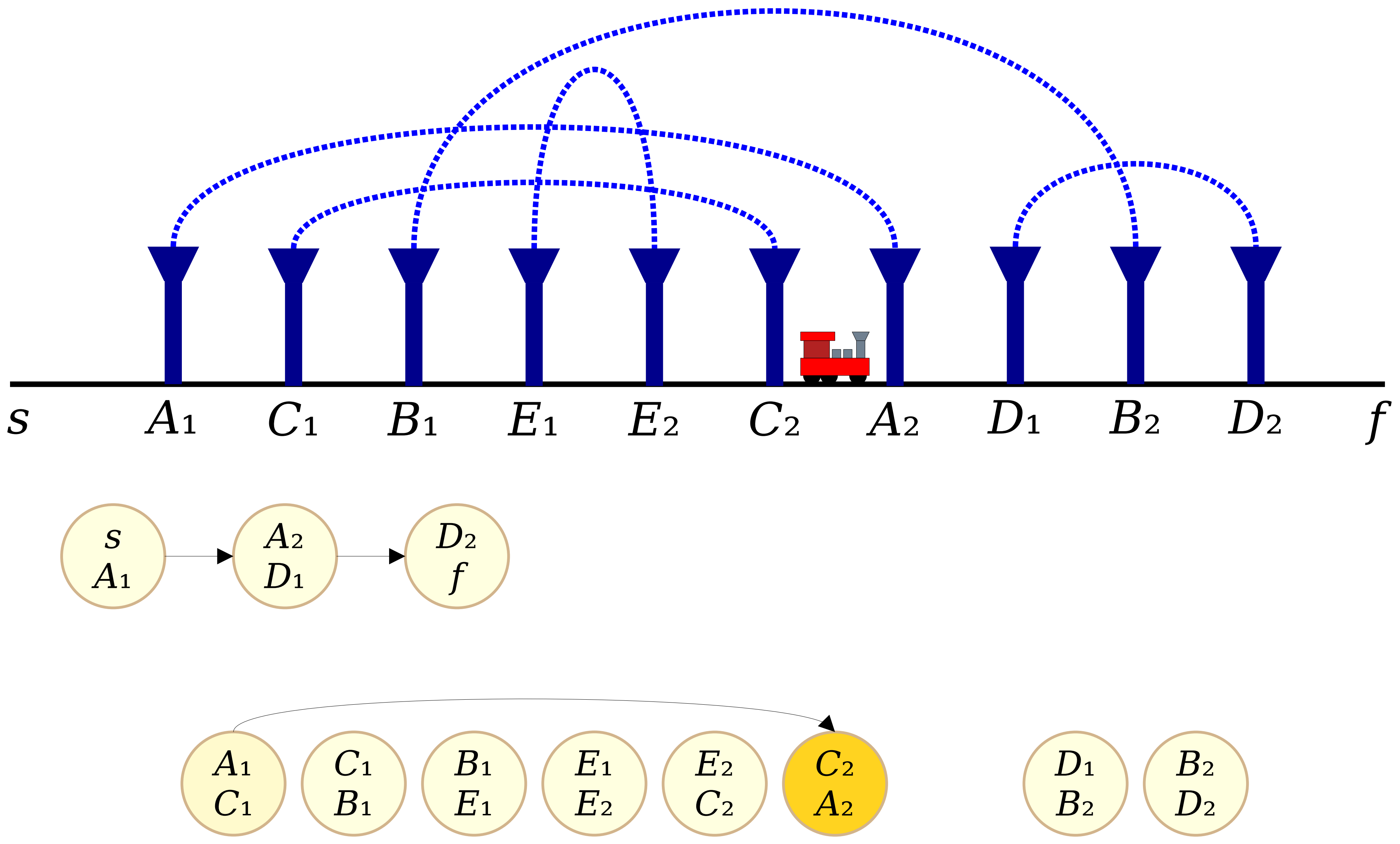
s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
C_2 A_2
D_1 B_2
B_2 D_2

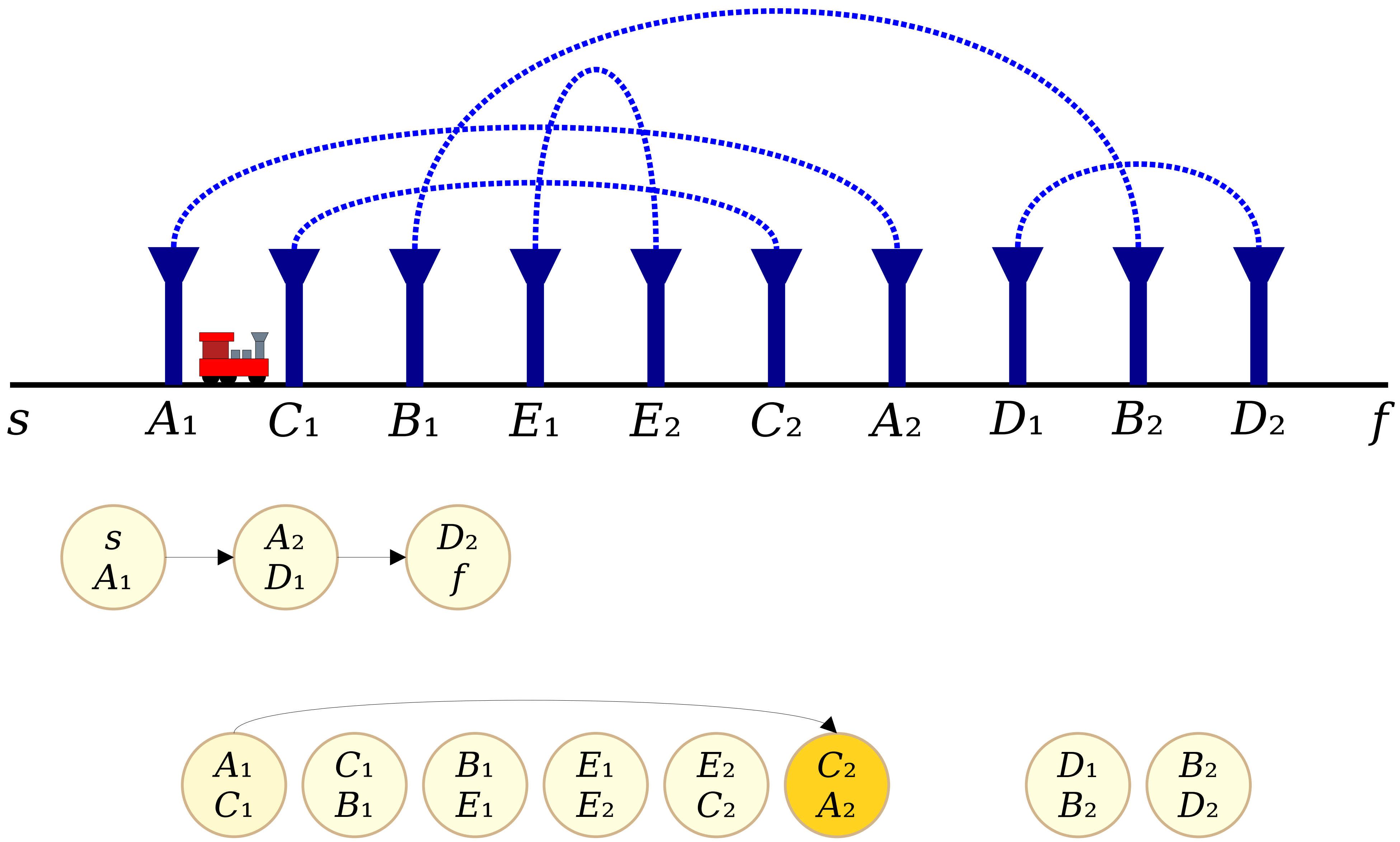


s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
C_2 A_2
D_1 B_2
B_2 D_2

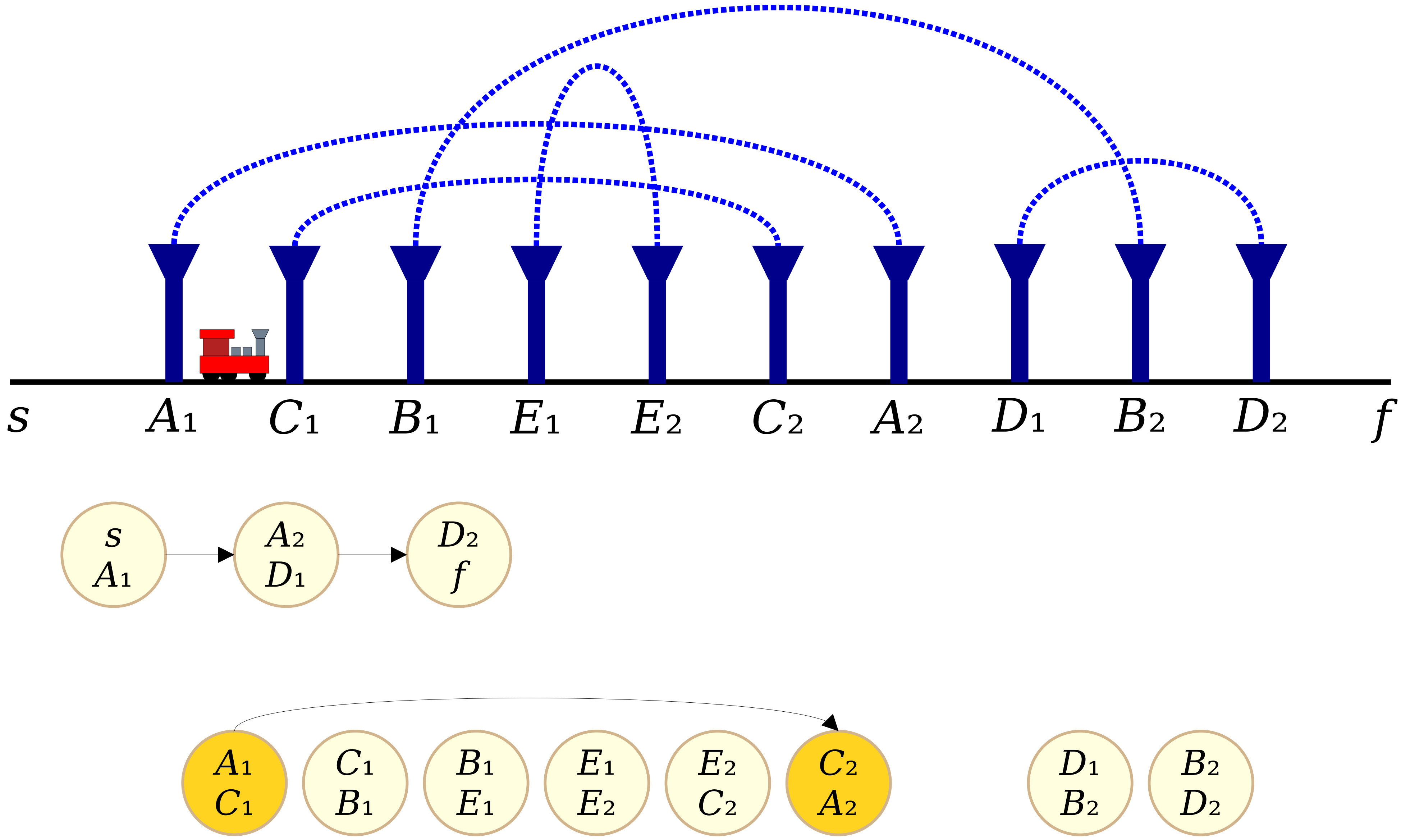


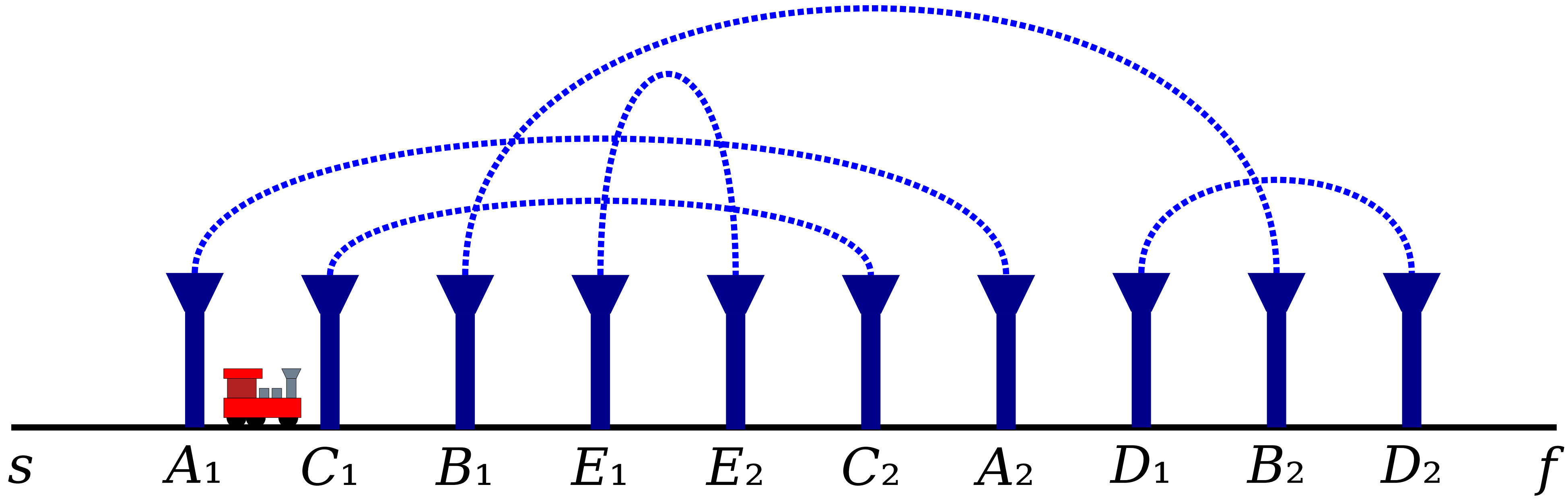
s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
C_2 A_2
D_1 B_2
B_2 D_2


s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C1 B1
B1 E1
E1 E2
E2 C2
C2 A2
D1 B2
B2 D2

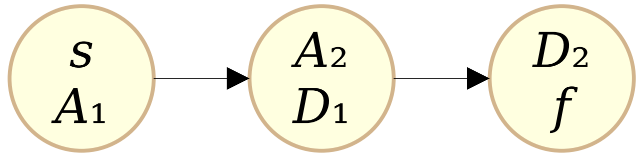


s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C1 B1
B1 E1
E1 E2
E2 C2
C2 A2
D1 B2
B2 D2

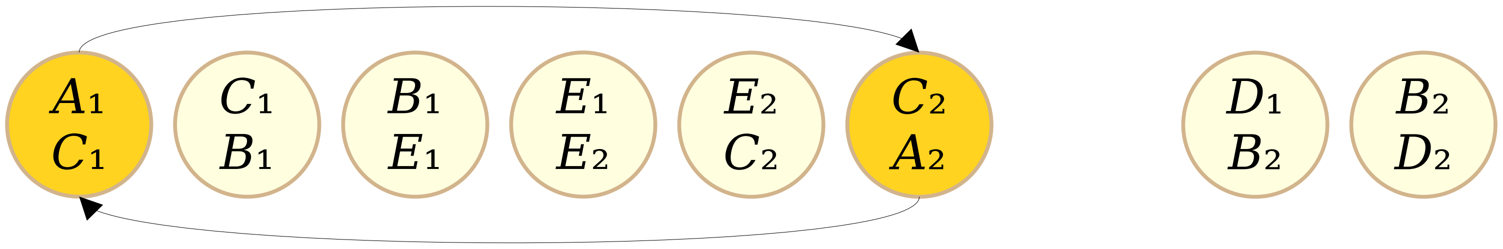

s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C1 B1
B1 E1
E1 E2
E2 C2
C2 A2
D1 B2
B2 D2



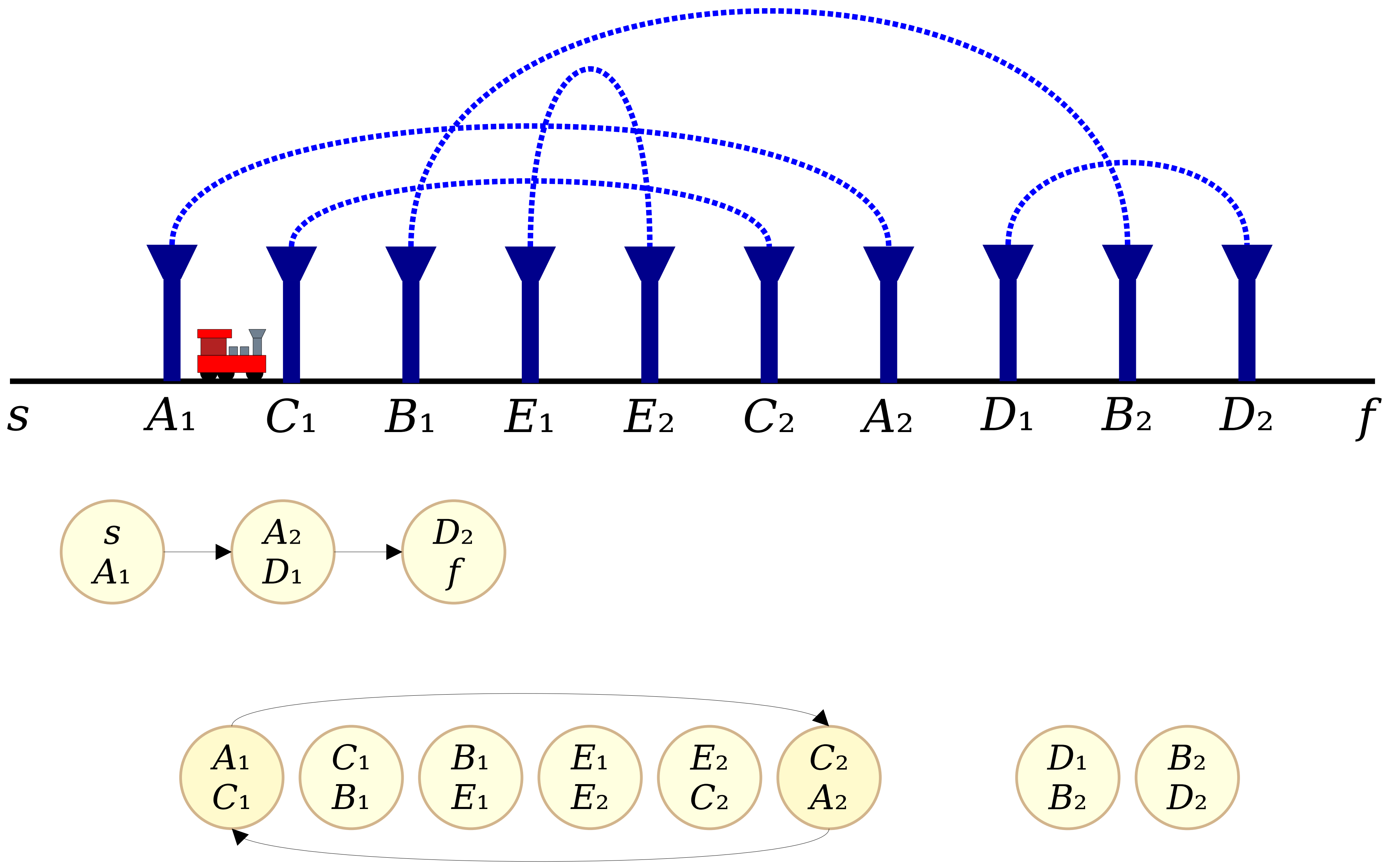
s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f



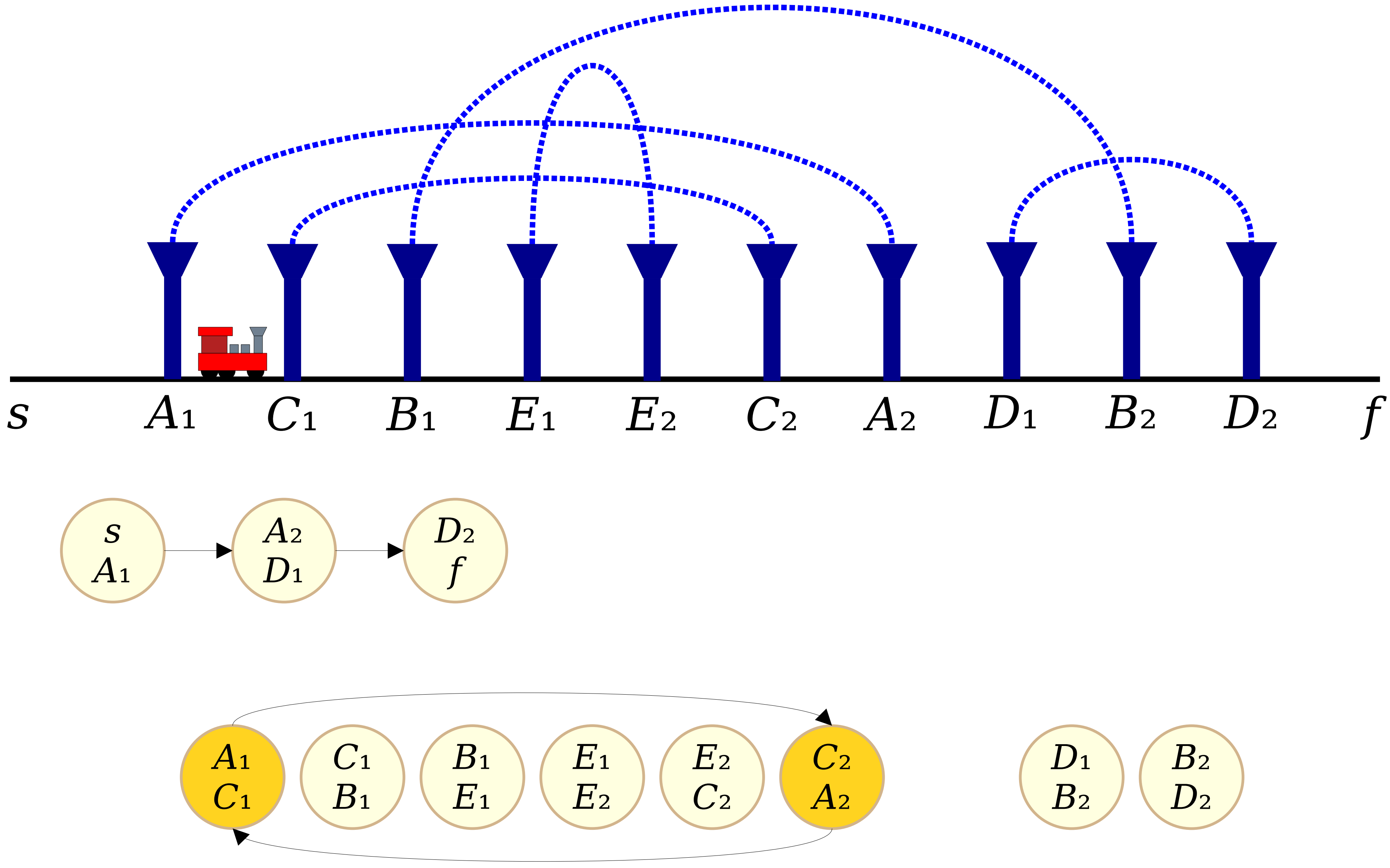
s A_1
A_2 D_1
D_2 f



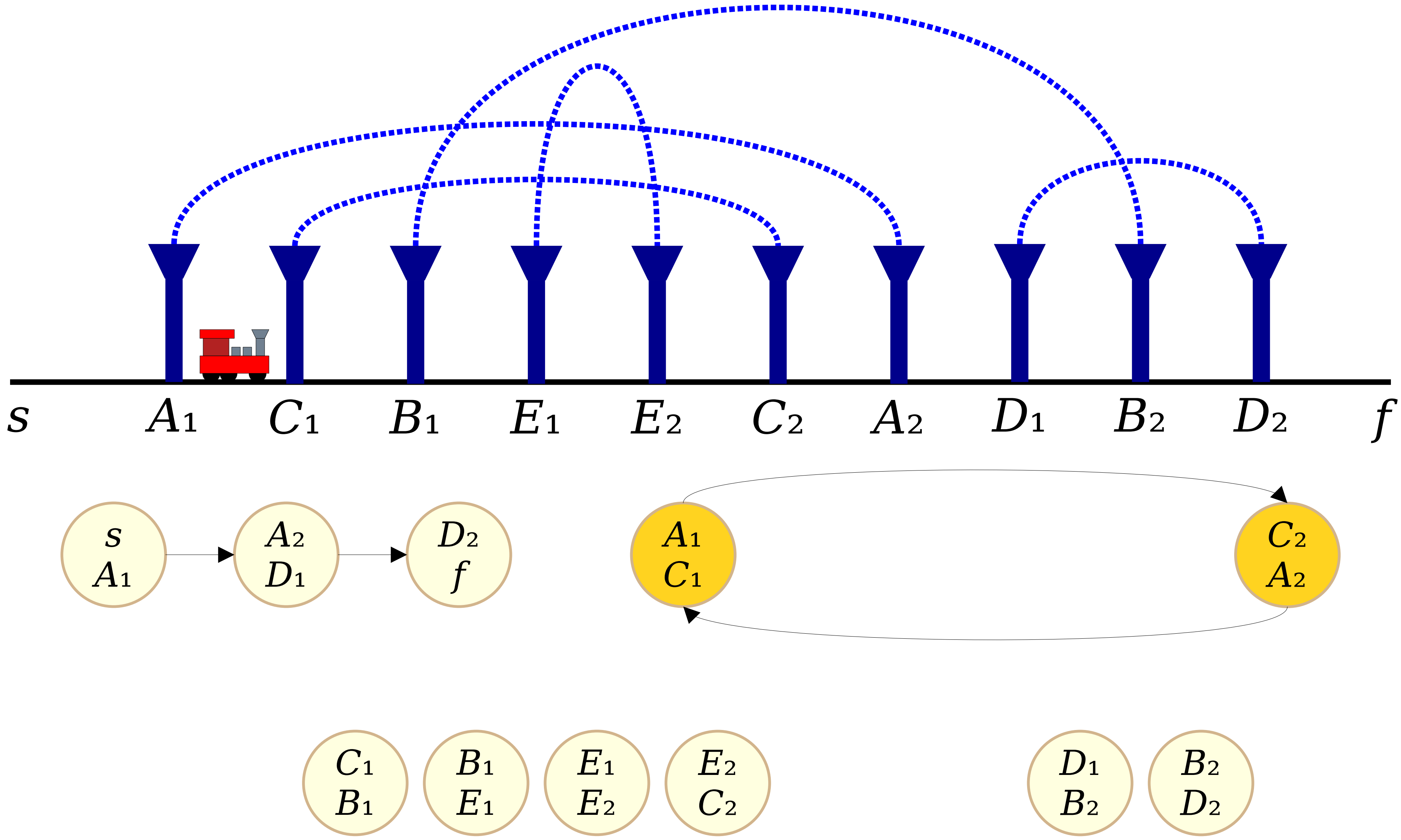
A_1 C_1
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
C_2 A_2
D_1 B_2
B_2 D_2



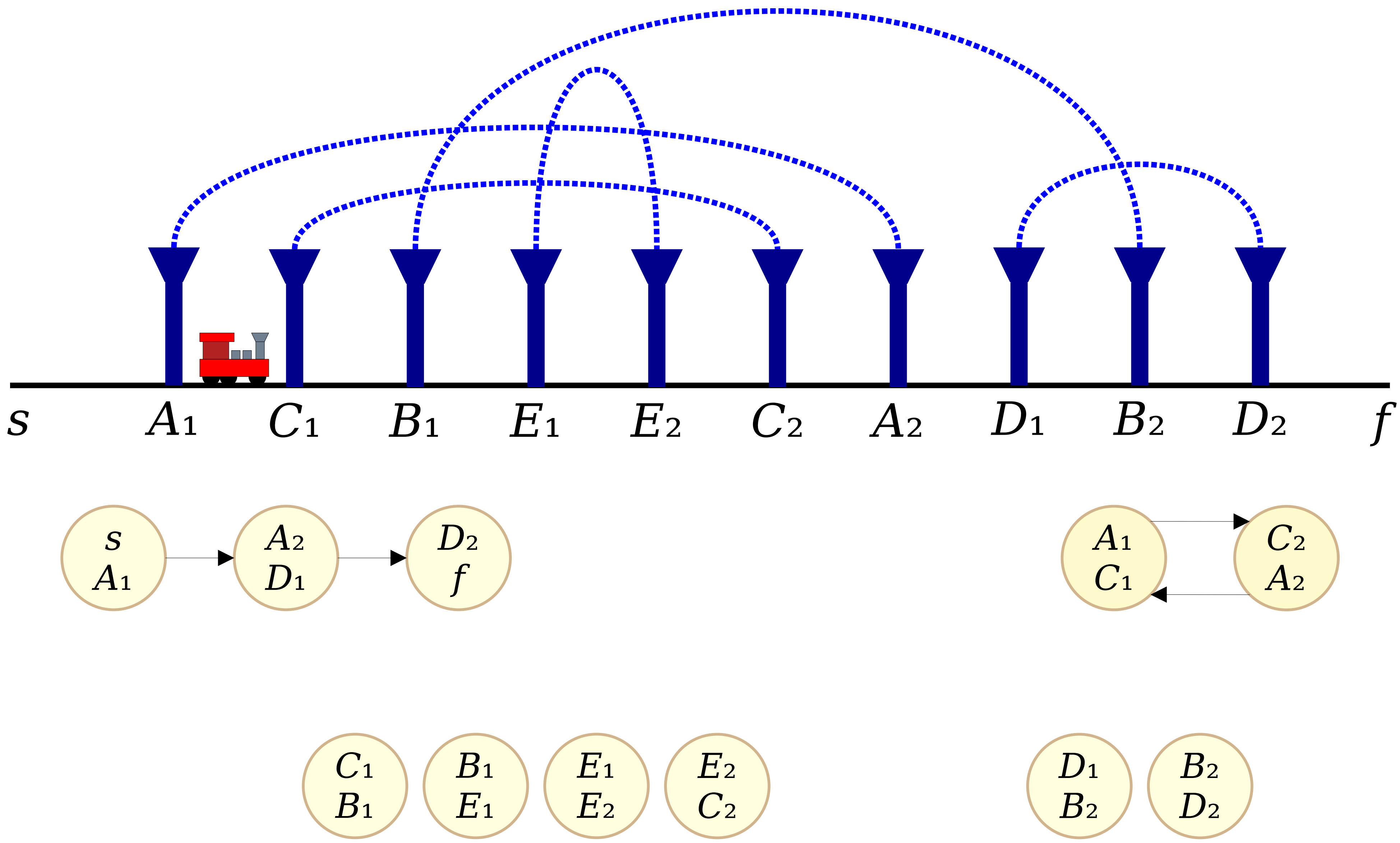
s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C1 B1
B1 E1
E1 E2
E2 C2
C2 A2
D1 B2
B2 D2

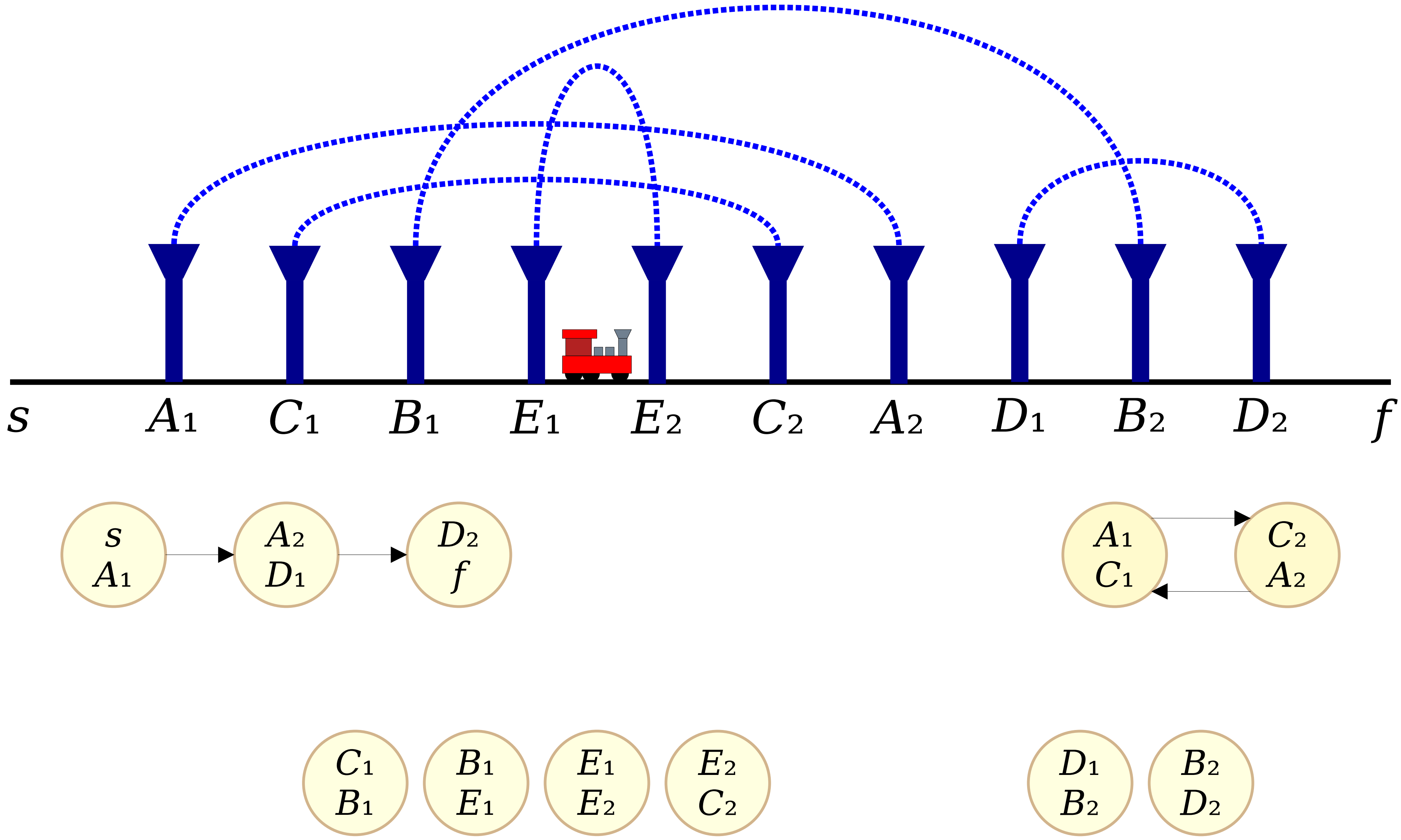


s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C1 B1
B1 E1
E1 E2
E2 C2
C2 A2
D1 B2
B2 D2

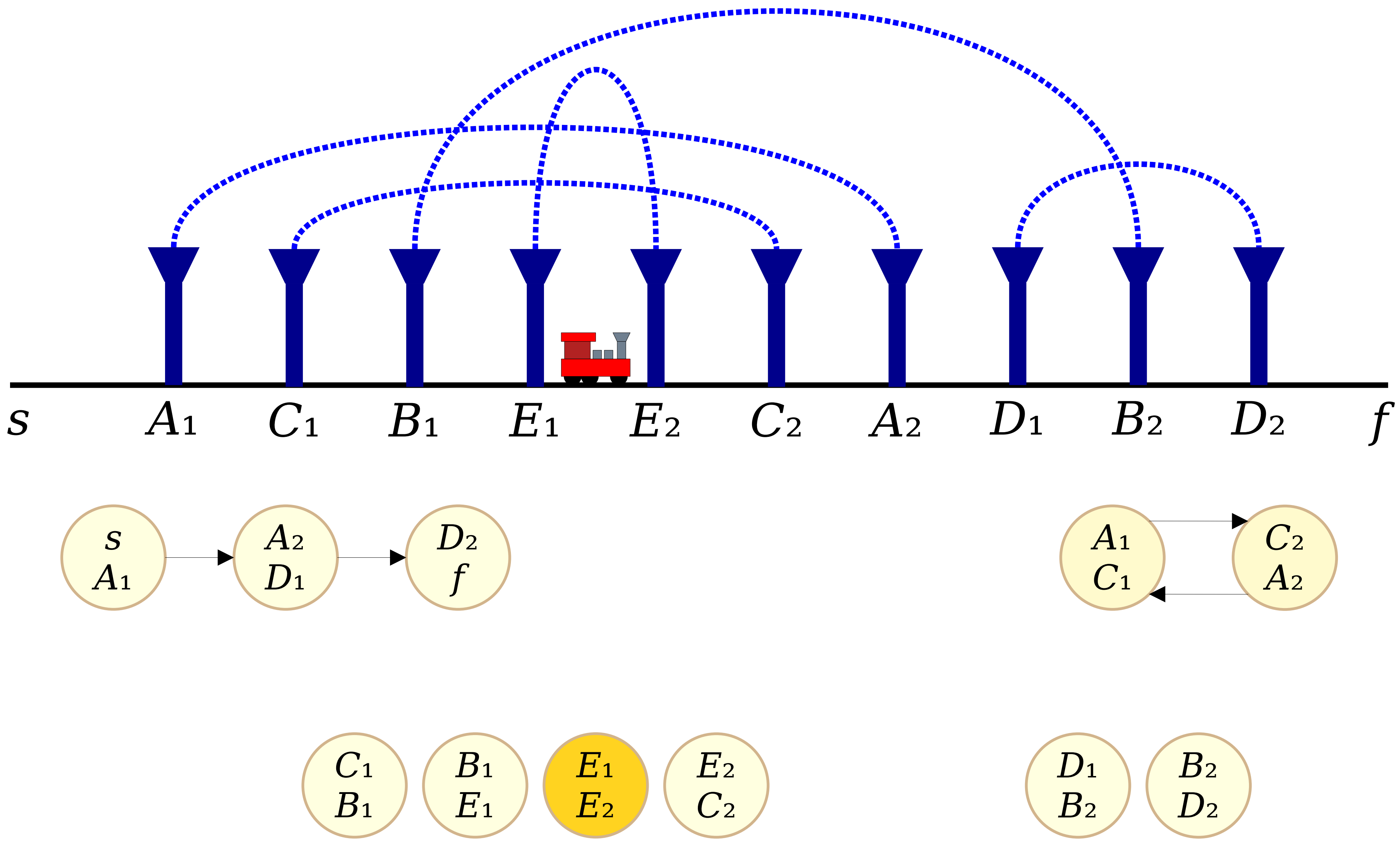


s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_2 A_2
C_1 B_1
B_1 E_1
E_1 E_2
E_2 C_2
D_1 B_2
B_2 D_2

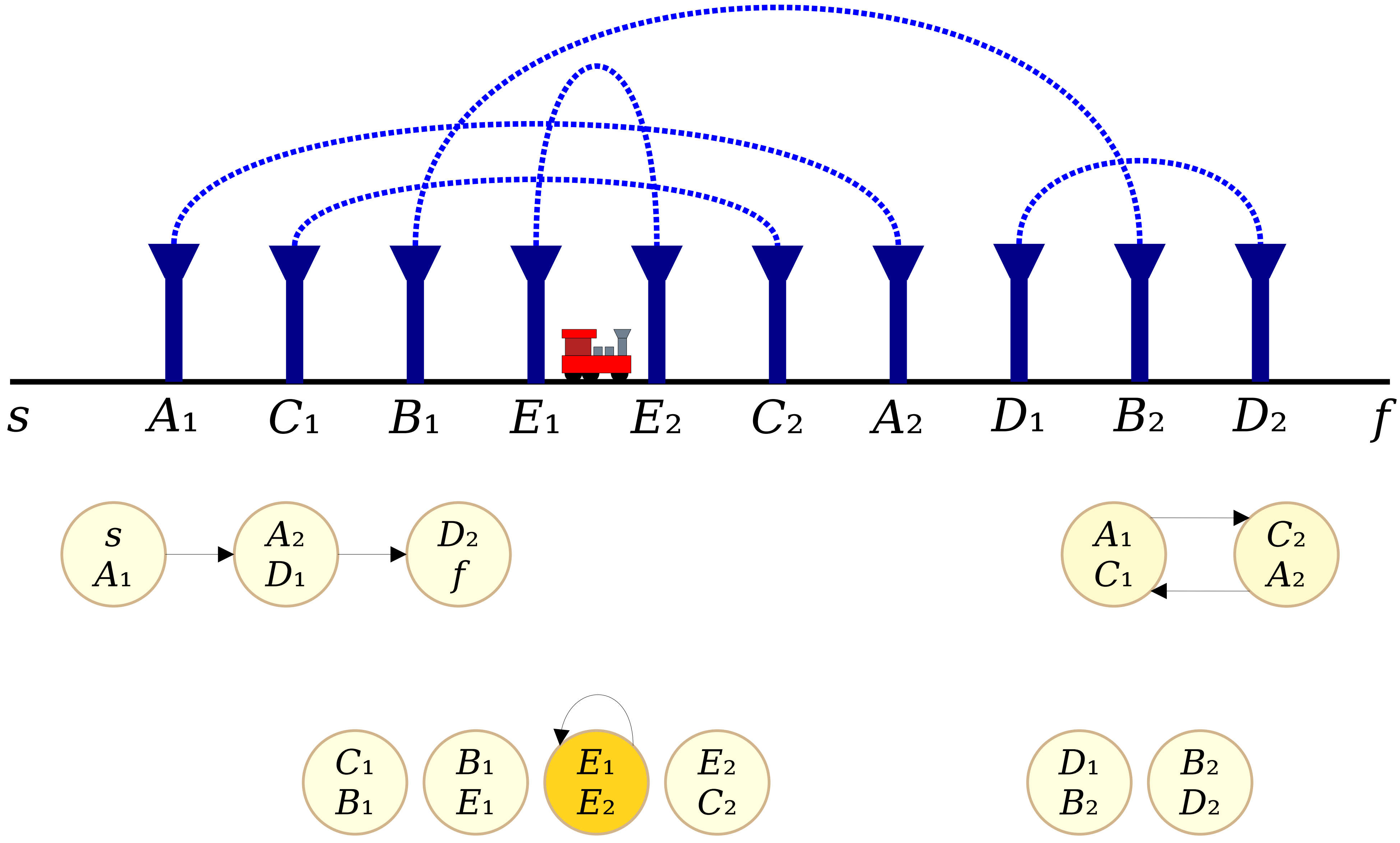

s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B1 E1
E1 E2
E2 C2
D1 B2
B2 D2



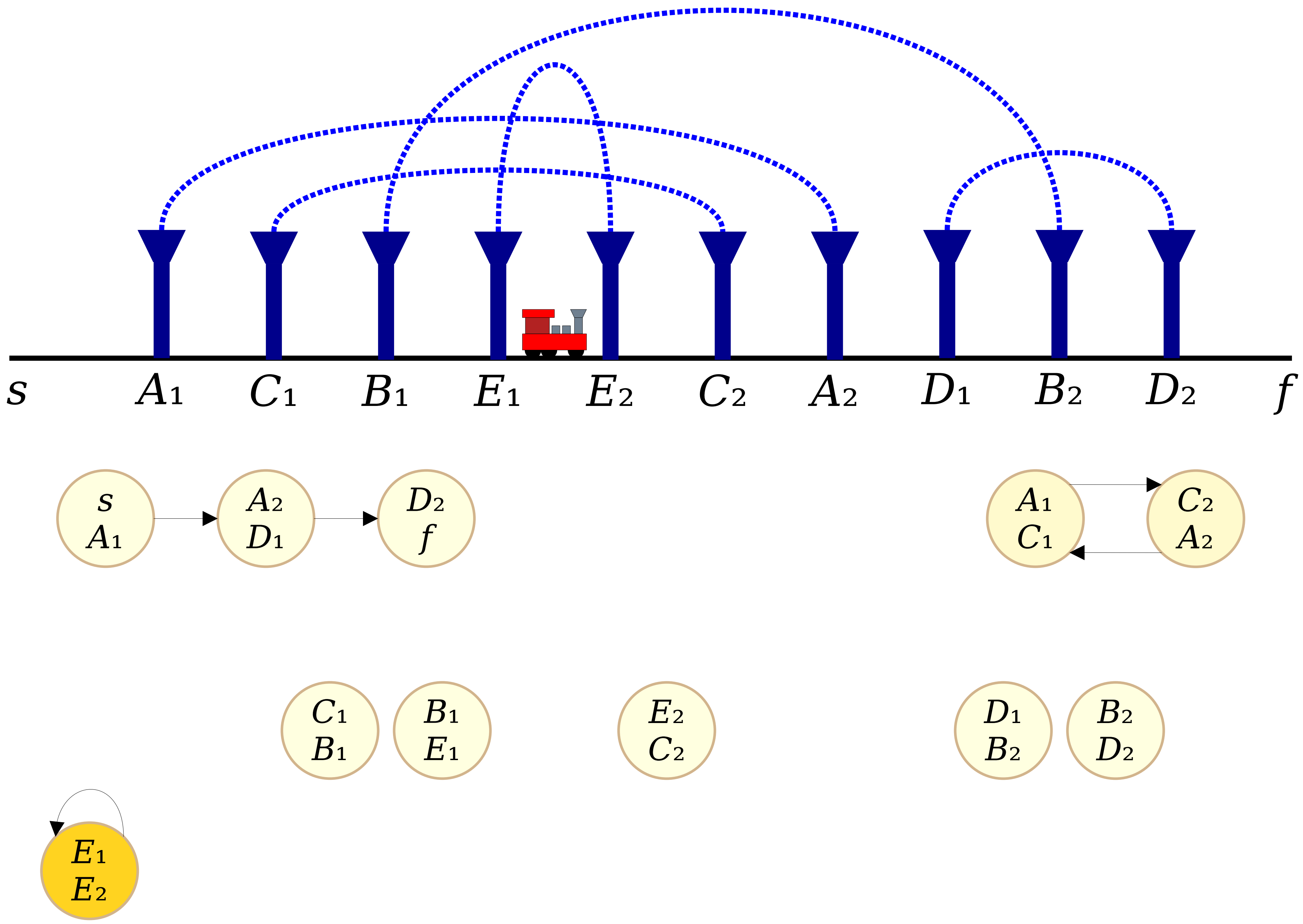
s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B1 E1
E1 E2
E2 C2
D1 B2
B2 D2



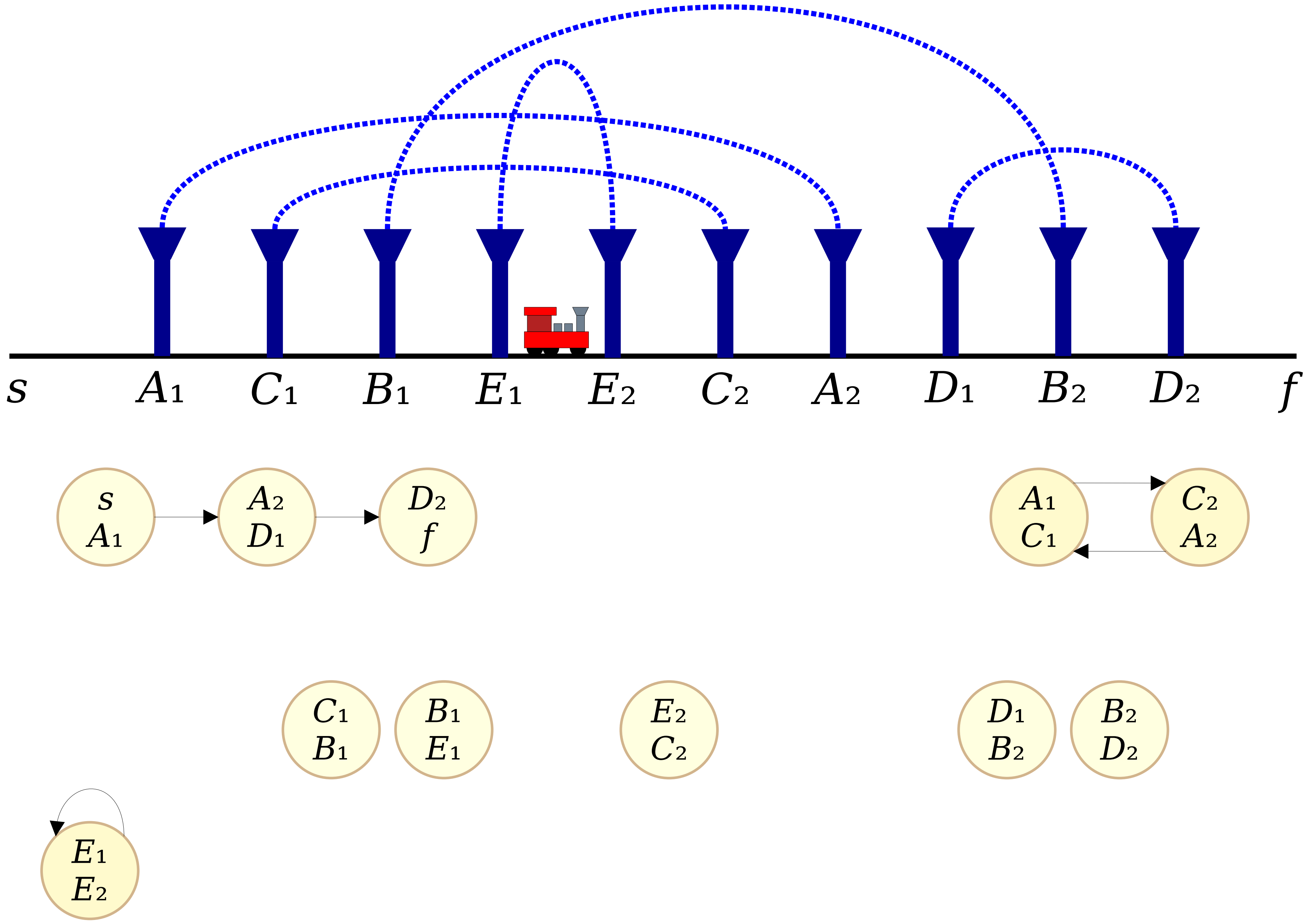
s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B1 E1
E1 E2
E2 C2
D1 B2
B2 D2



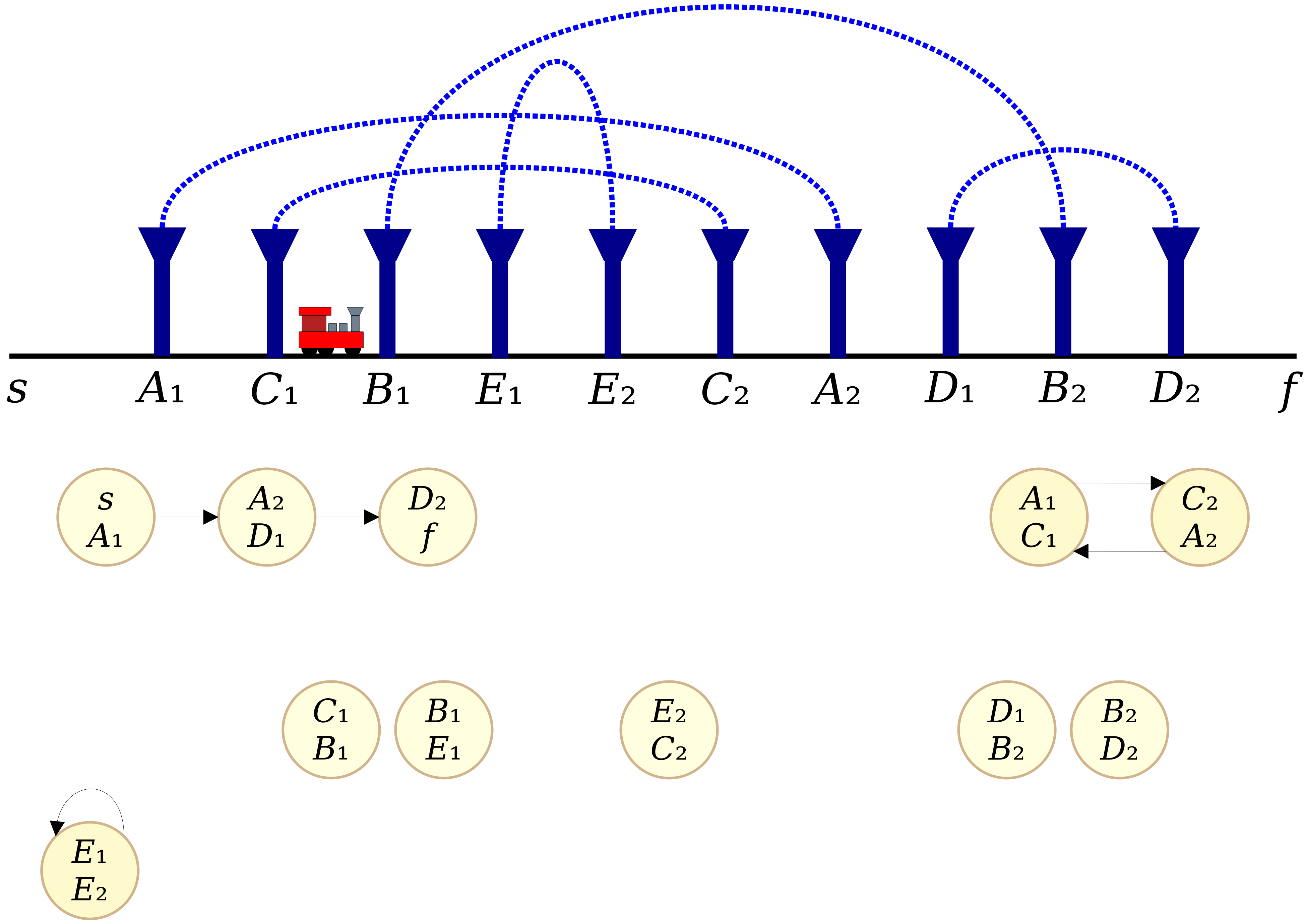
s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B1 E1
E1 E2
E2 C2
D1 B2
B2 D2



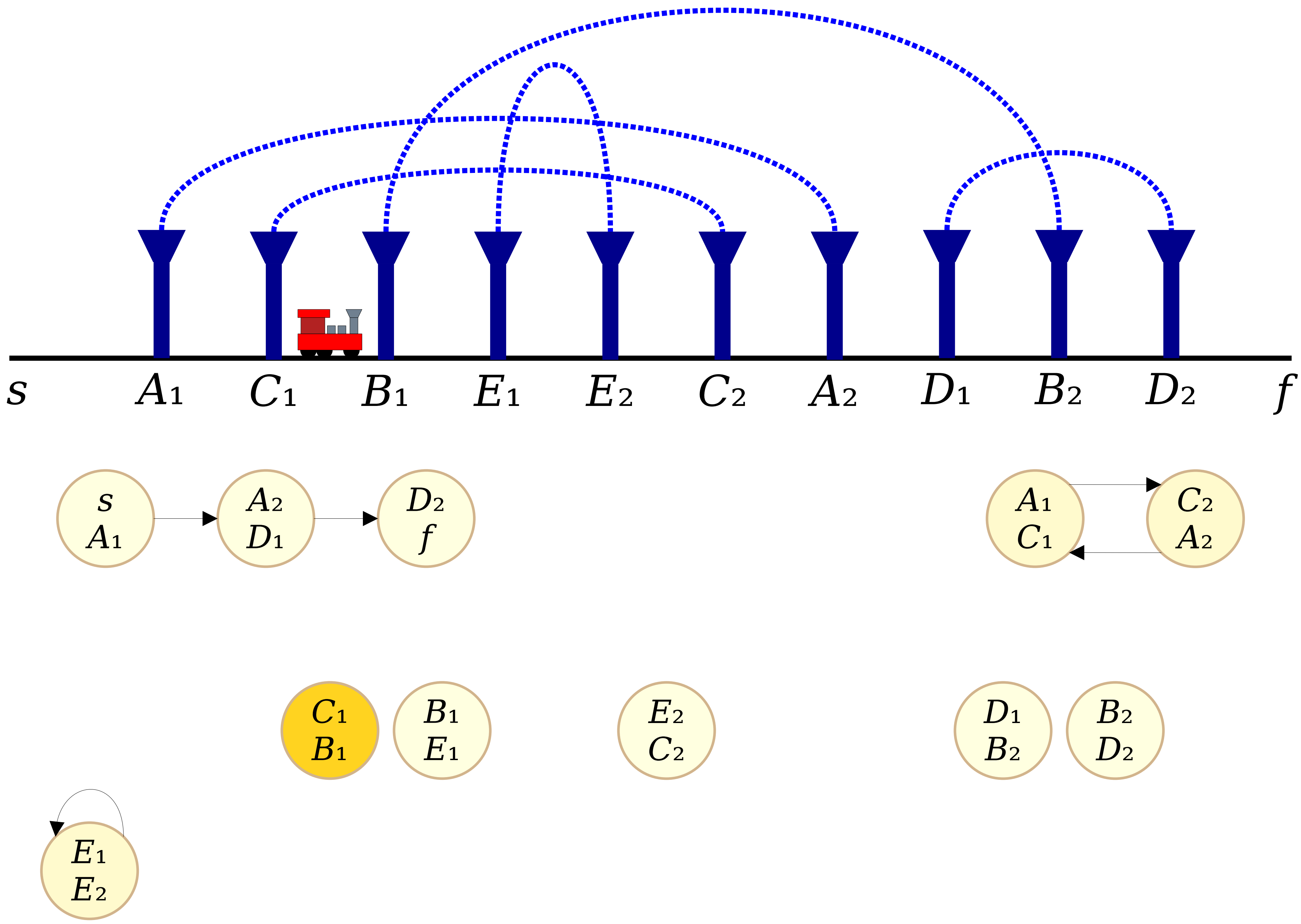
s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_2 A_2
C_1 B_1
B_1 E_1
E_2 C_2
D_1 B_2
B_2 D_2
E_1 E_2

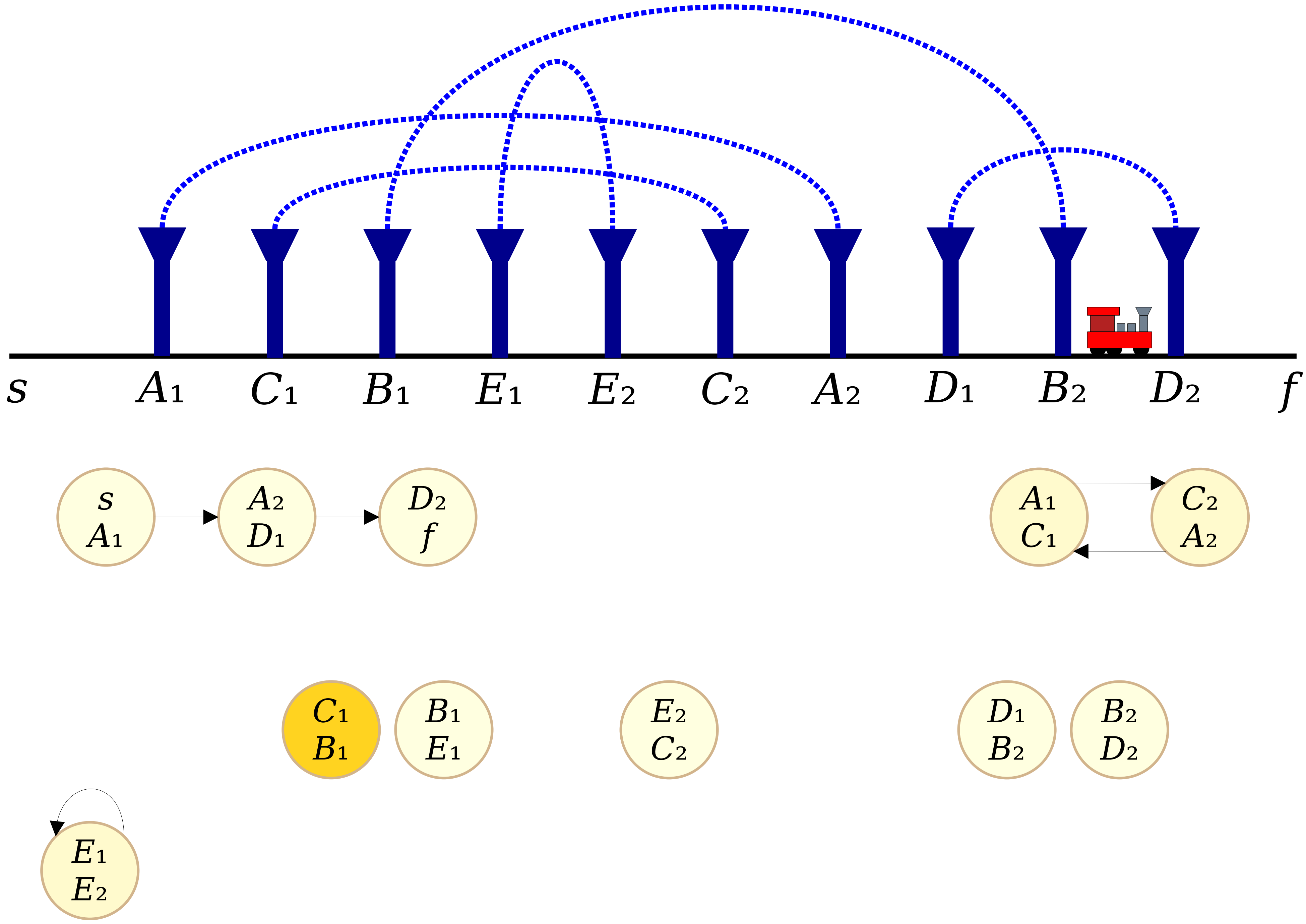


s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_2 A_2
C_1 B_1
B_1 E_1
E_2 C_2
D_1 B_2
B_2 D_2
E_1 E_2

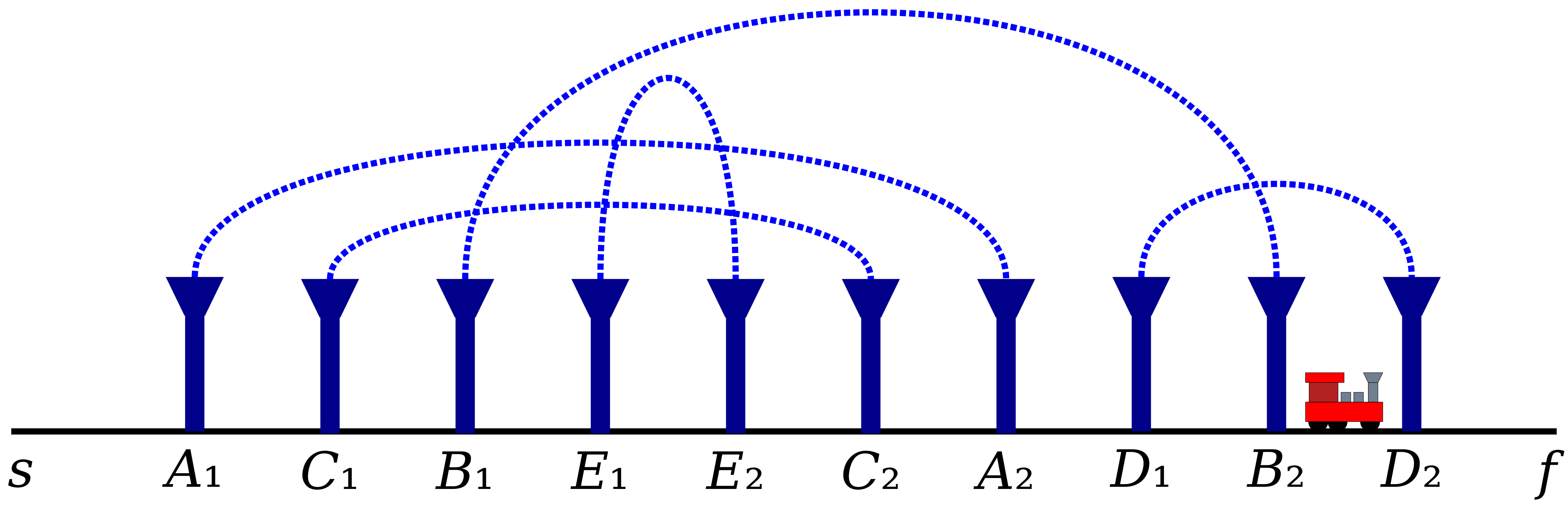


s
A₁
C₁
B₁
E₁
E₂
C₂
A₂
D₁
B₂
D₂
f
s A₁
A₂ D₁
D₂ f
A₁ C₁
C₂ A₂
C₁ B₁
B₁ E₁
E₂ C₂
D₁ B₂
B₂ D₂
E₁ E₂

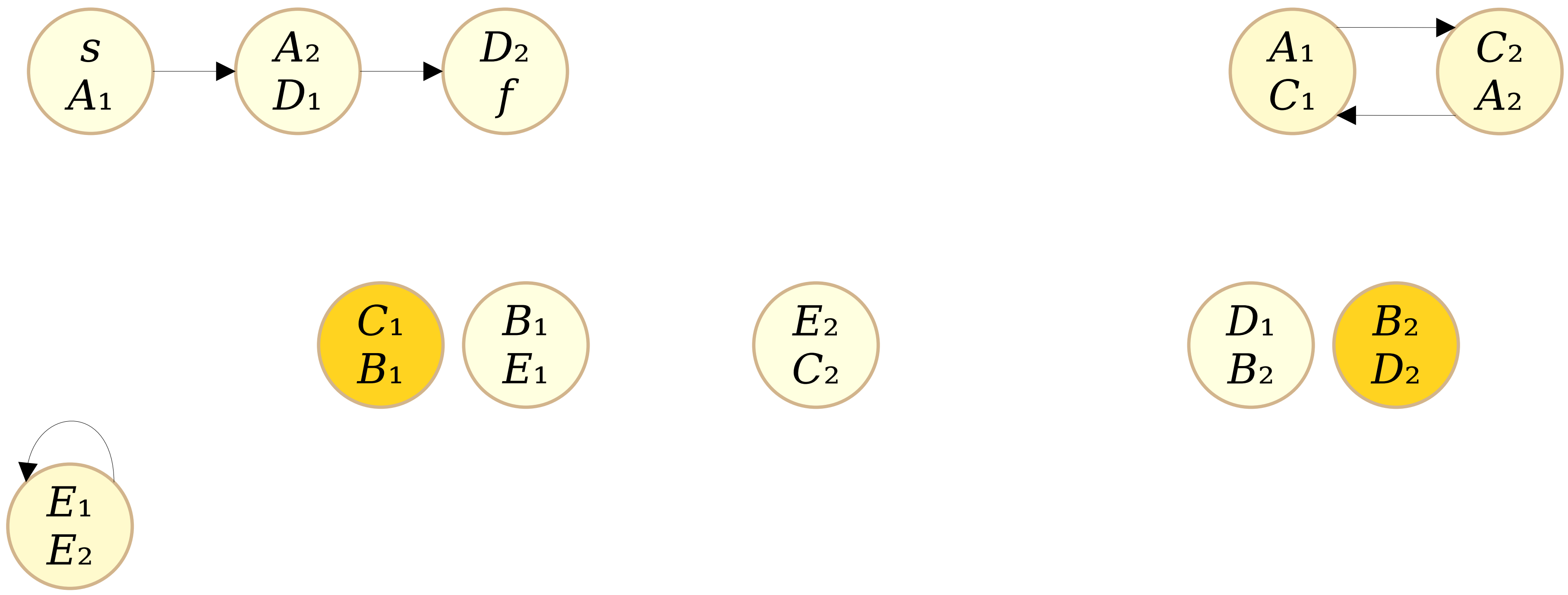

s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_2 A_2
C_1 B_1
B_1 E_1
E_2 C_2
D_1 B_2
B_2 D_2
E_1 E_2



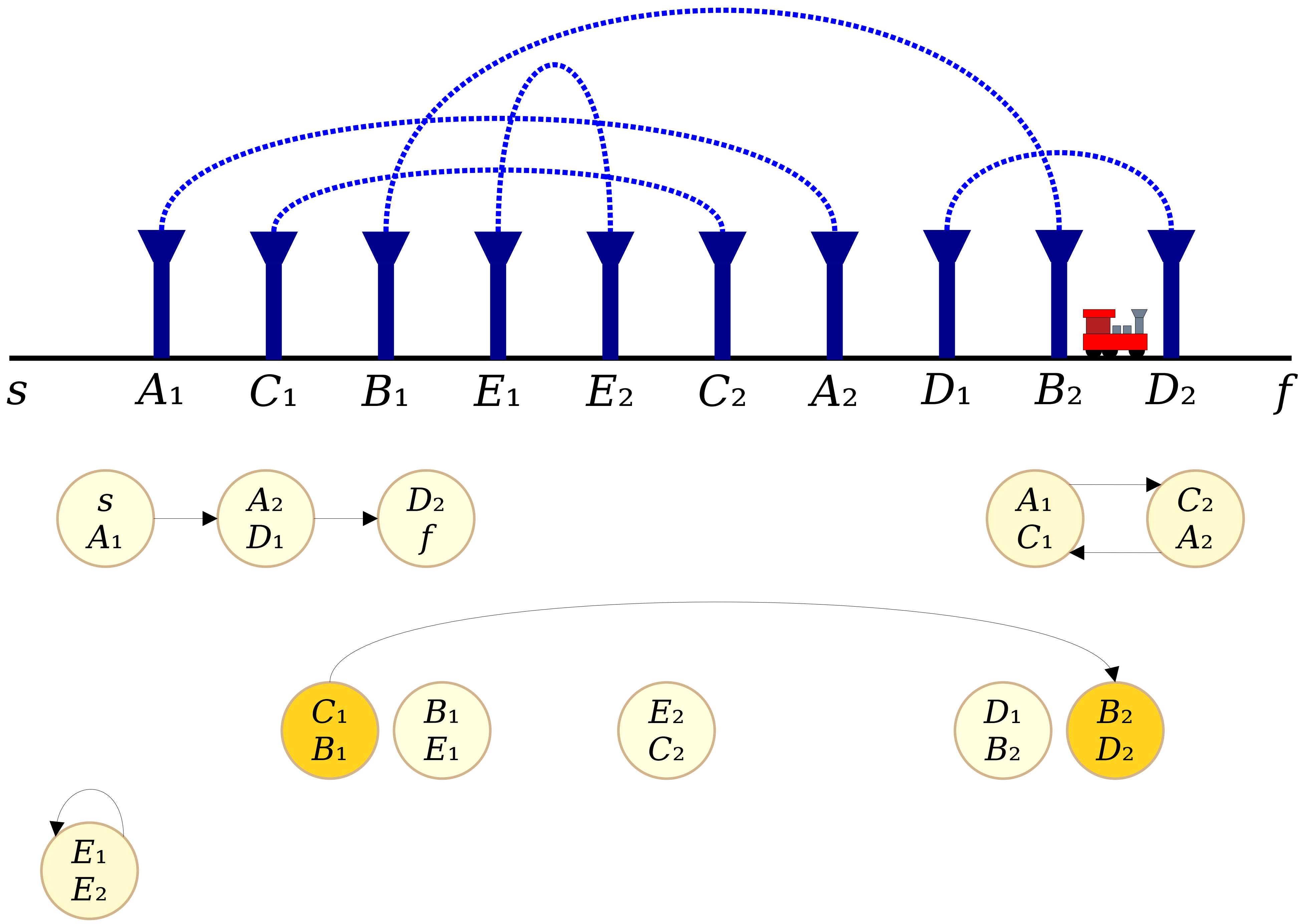
s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_2 A_2
C_1 B_1
B_1 E_1
E_2 C_2
D_1 B_2
B_2 D_2
E_1 E_2



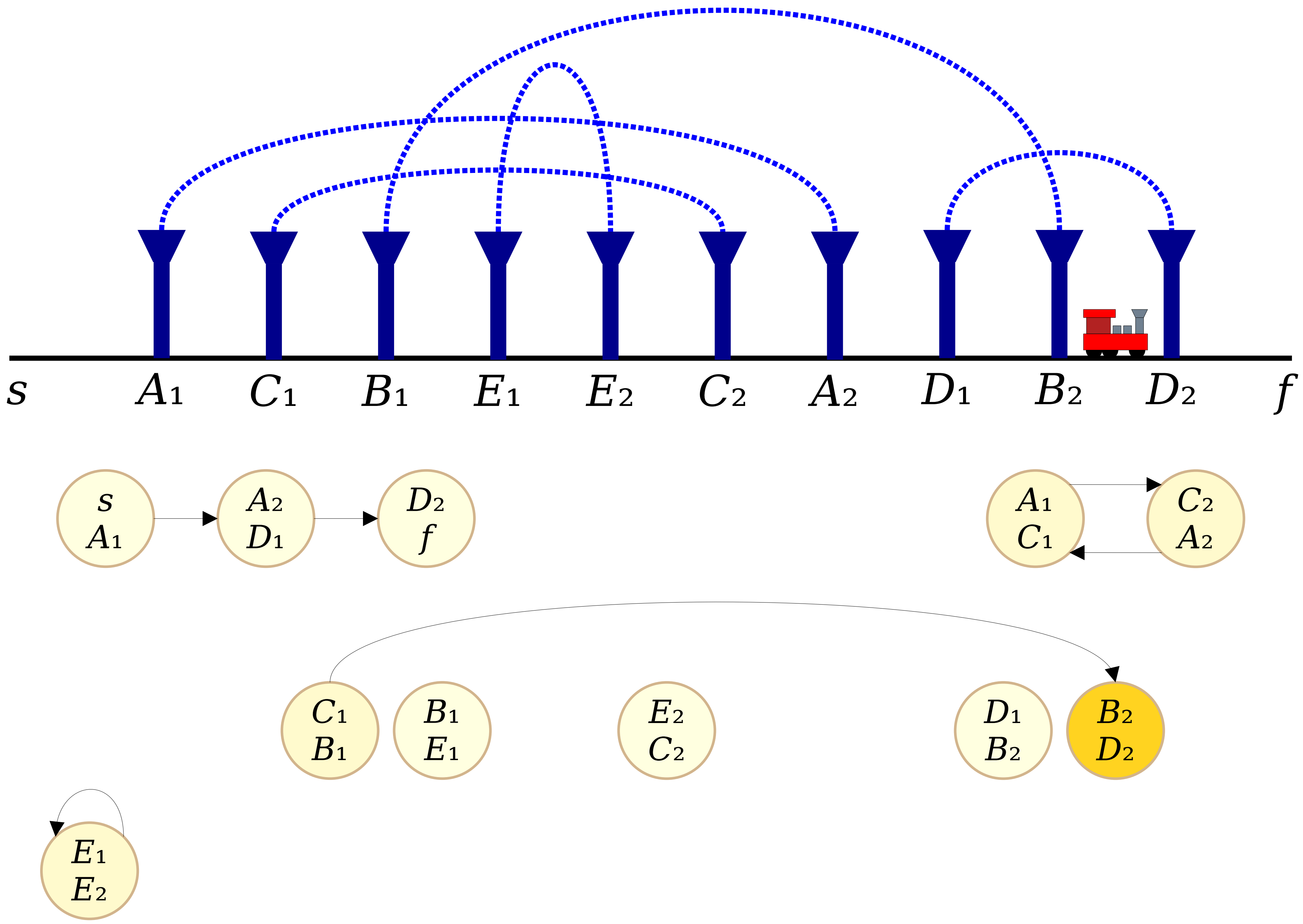
s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f

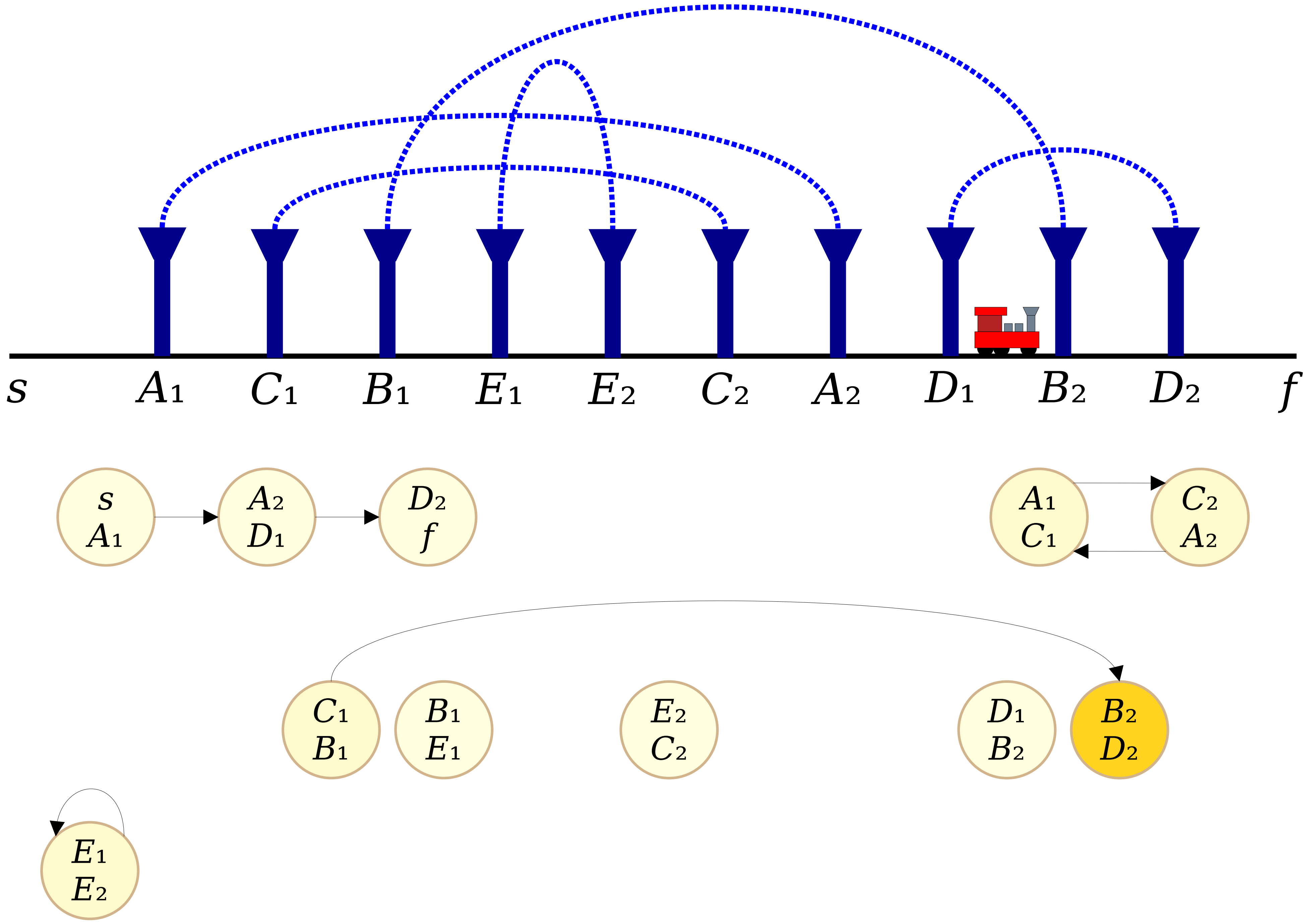


s A_1
A_2 D_1
D_2 f
A_1 C_1
C_2 A_2
C_1 B_1
B_1 E_1
E_2 C_2
D_1 B_2
B_2 D_2
E_1 E_2

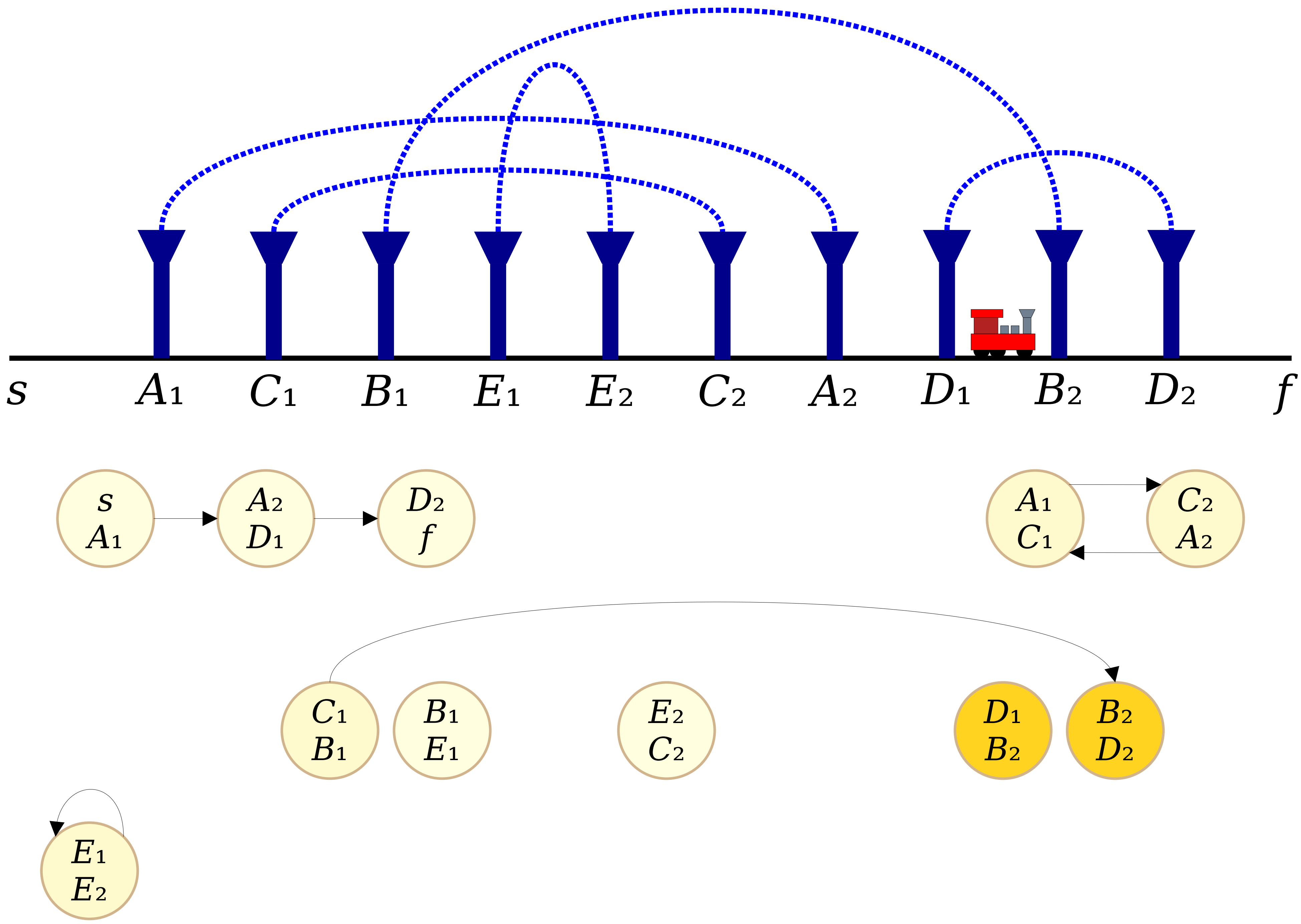


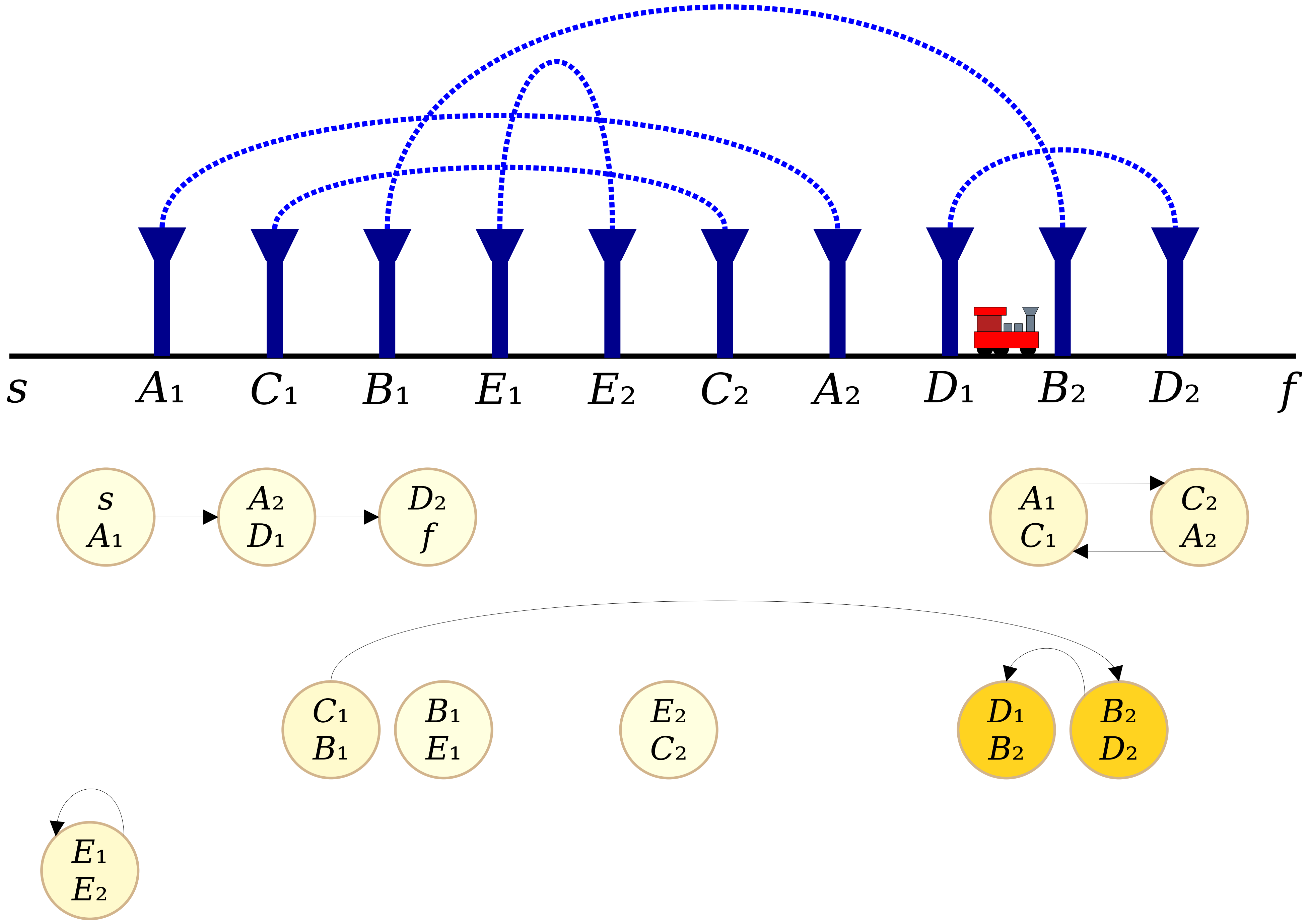
s A_1 C_1 B_1 E_1 E_2 C_2 A_2 D_1 B_2 D_2 f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_2 A_2
C_1 B_1
B_1 E_1
E_2 C_2
D_1 B_2
B_2 D_2
E_1 E_2


s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B1 E1
E2 C2
D1 B2
B2 D2
E1 E2

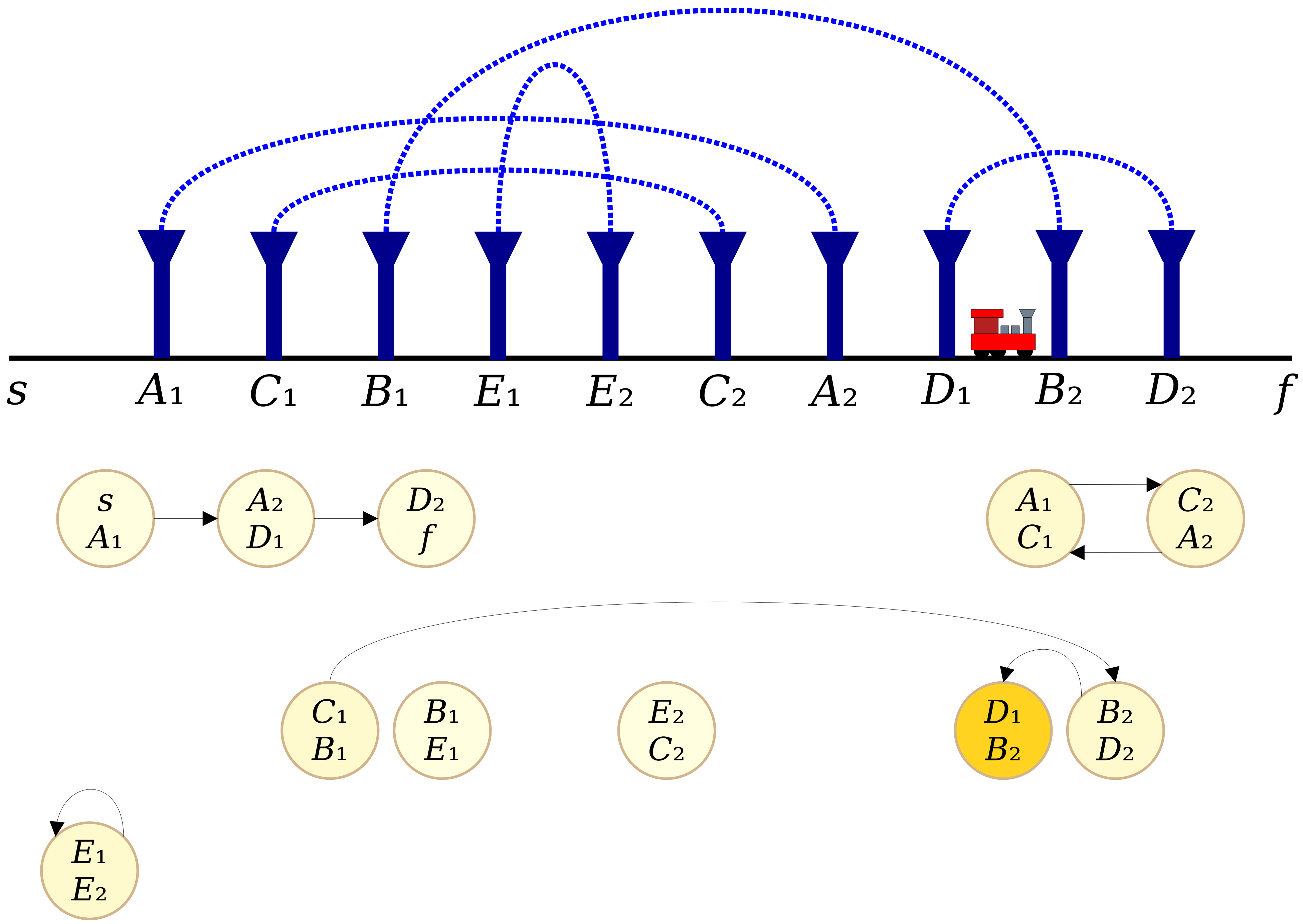


s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B1 E1
E2 C2
D1 B2
B2 D2
E1 E2

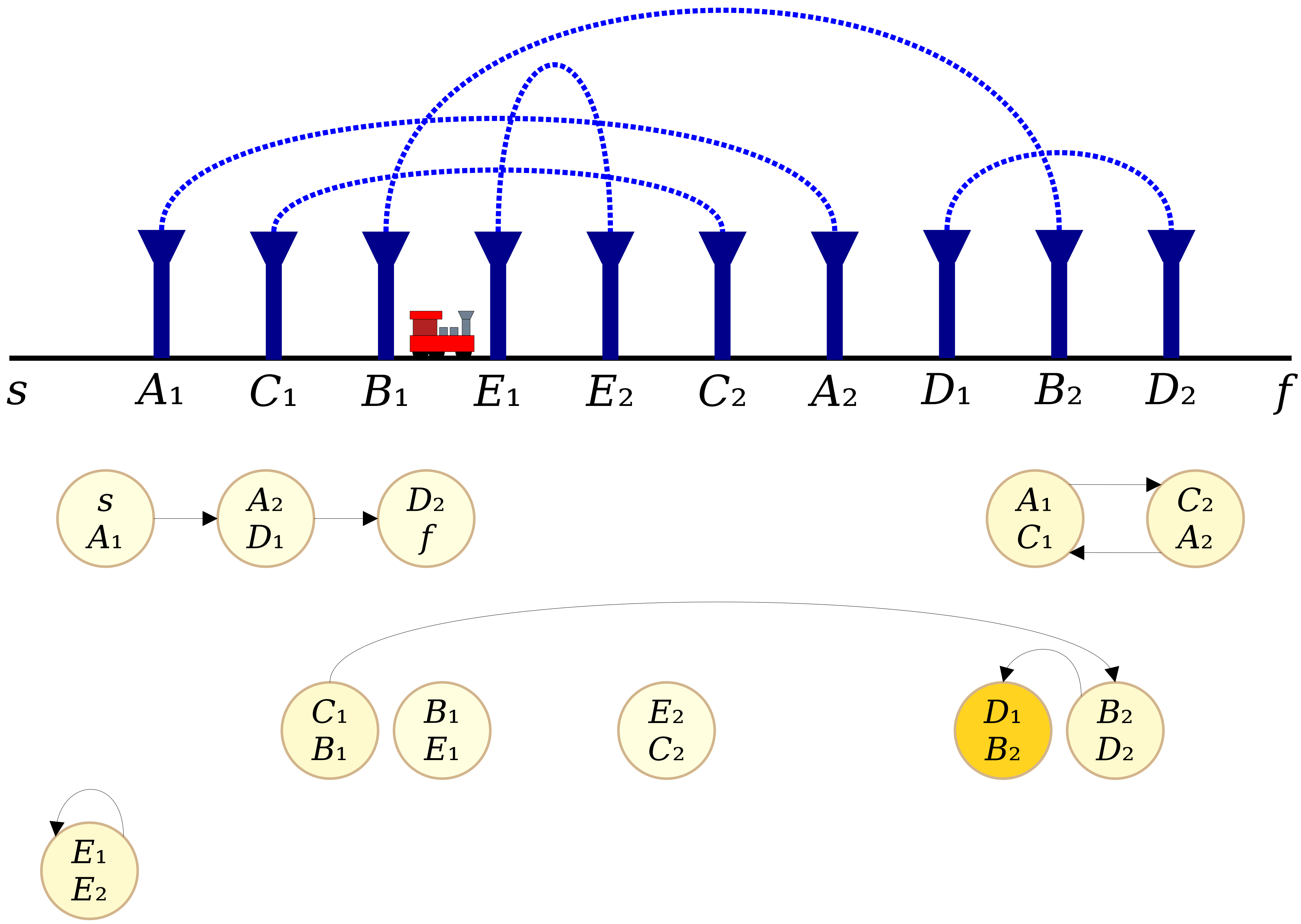

s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_2 A_2
C_1 B_1
B_1 E_1
E_2 C_2
D_1 B_2
B_2 D_2
E_1 E_2

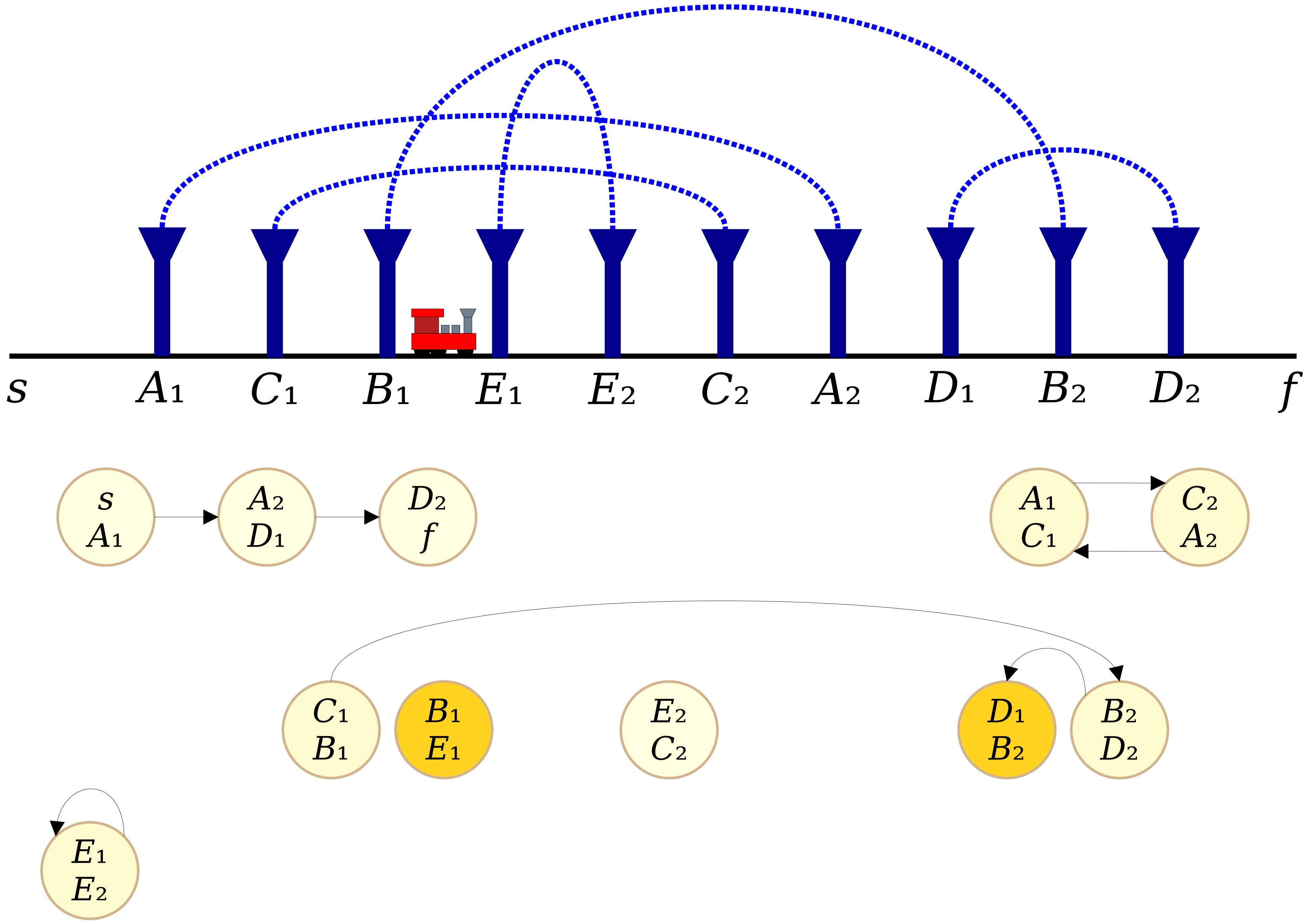


s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B1 E1
E2 C2
D1 B2
B2 D2
E1 E2

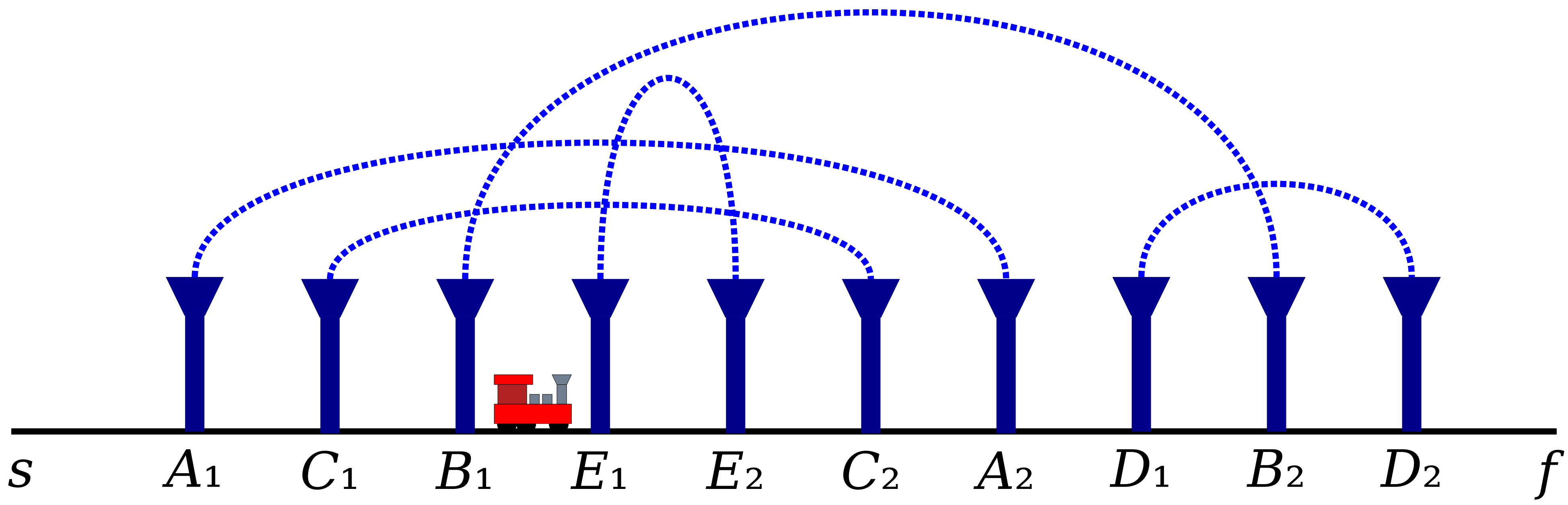


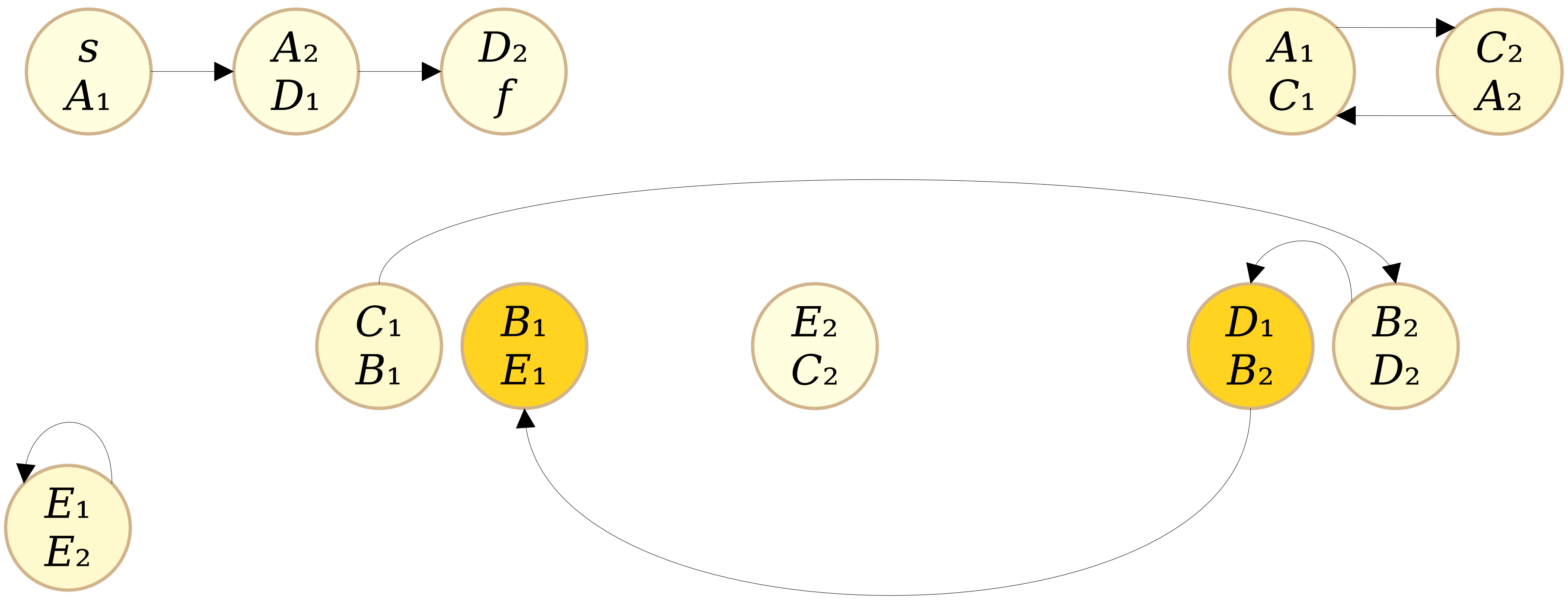
s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B1 E1
E2 C2
D1 B2
B2 D2
E1 E2

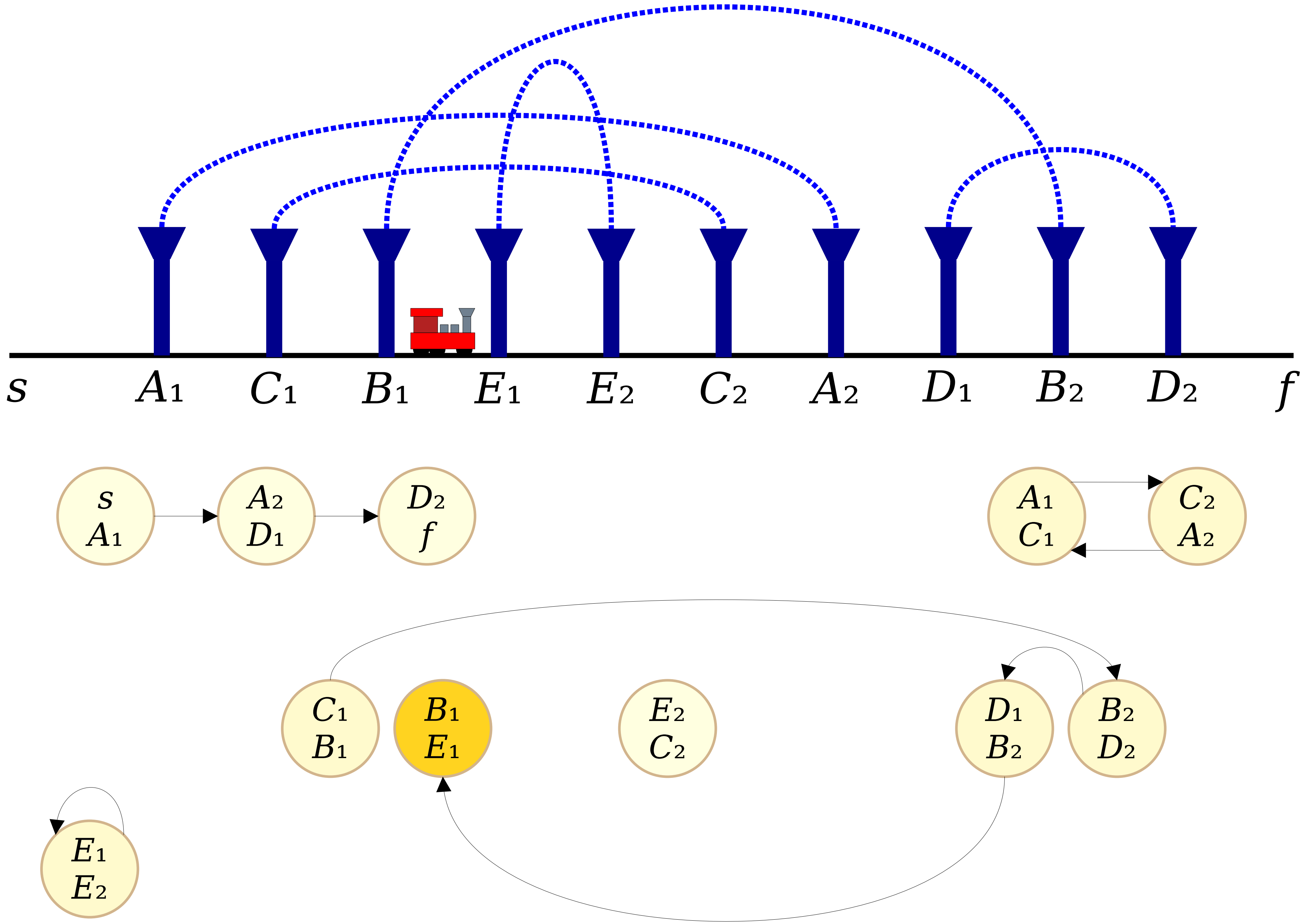

s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B1 E1
E2 C2
D1 B2
B2 D2
E1 E2

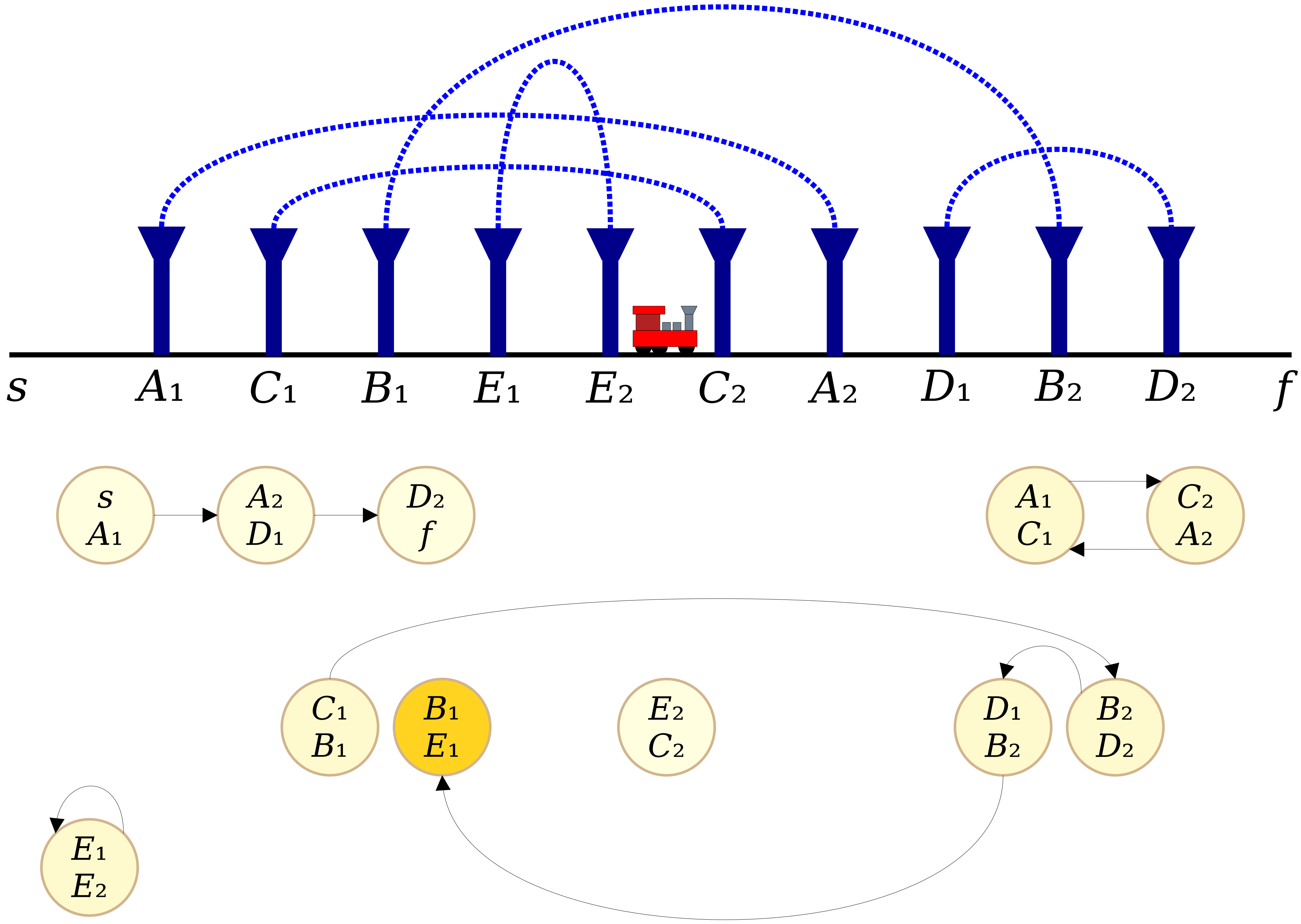


s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B1 E1
E2 C2
D1 B2
B2 D2
E1 E2

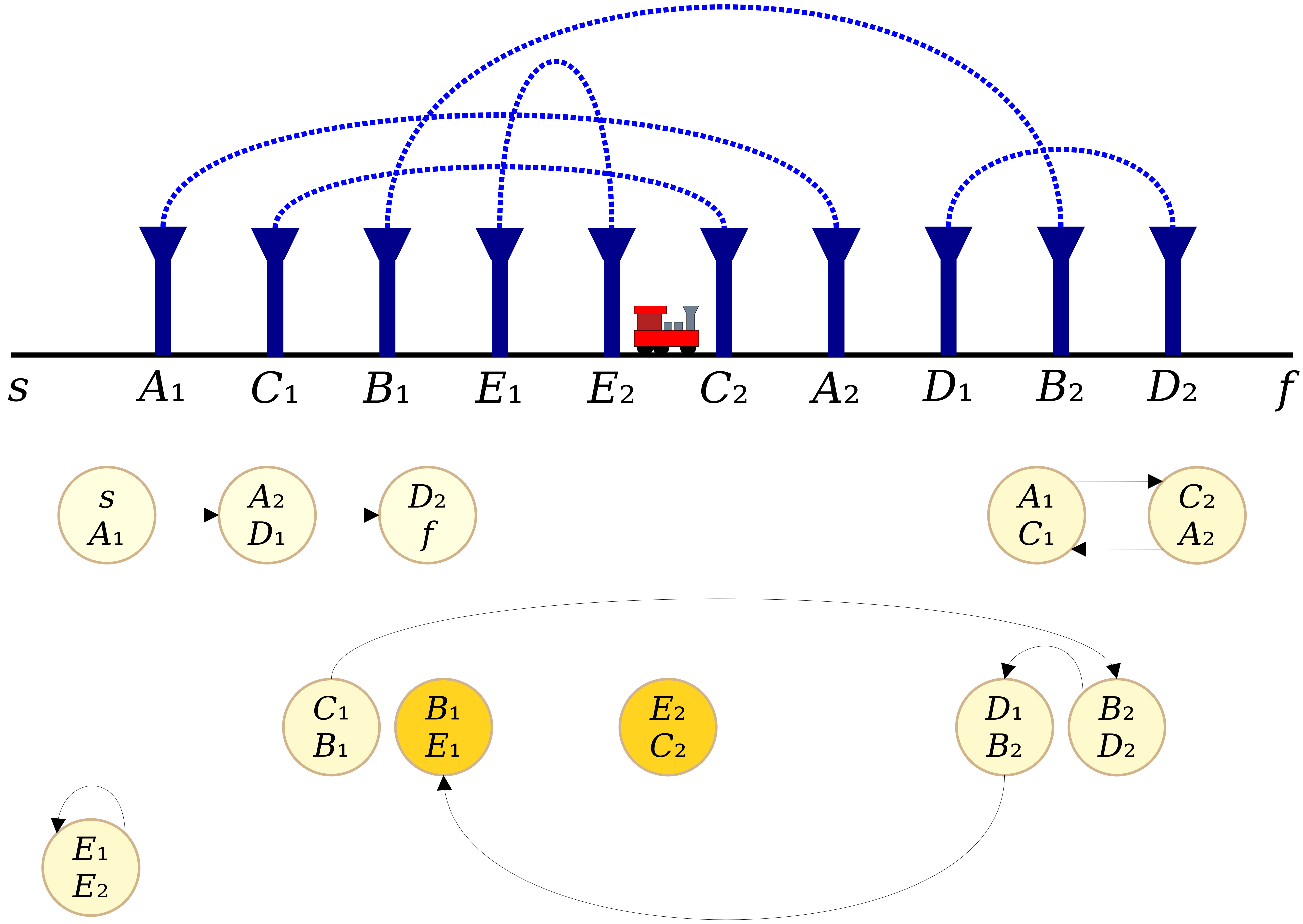

s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f


s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B1 E1
E2 C2
D1 B2
B2 D2
E1 E2

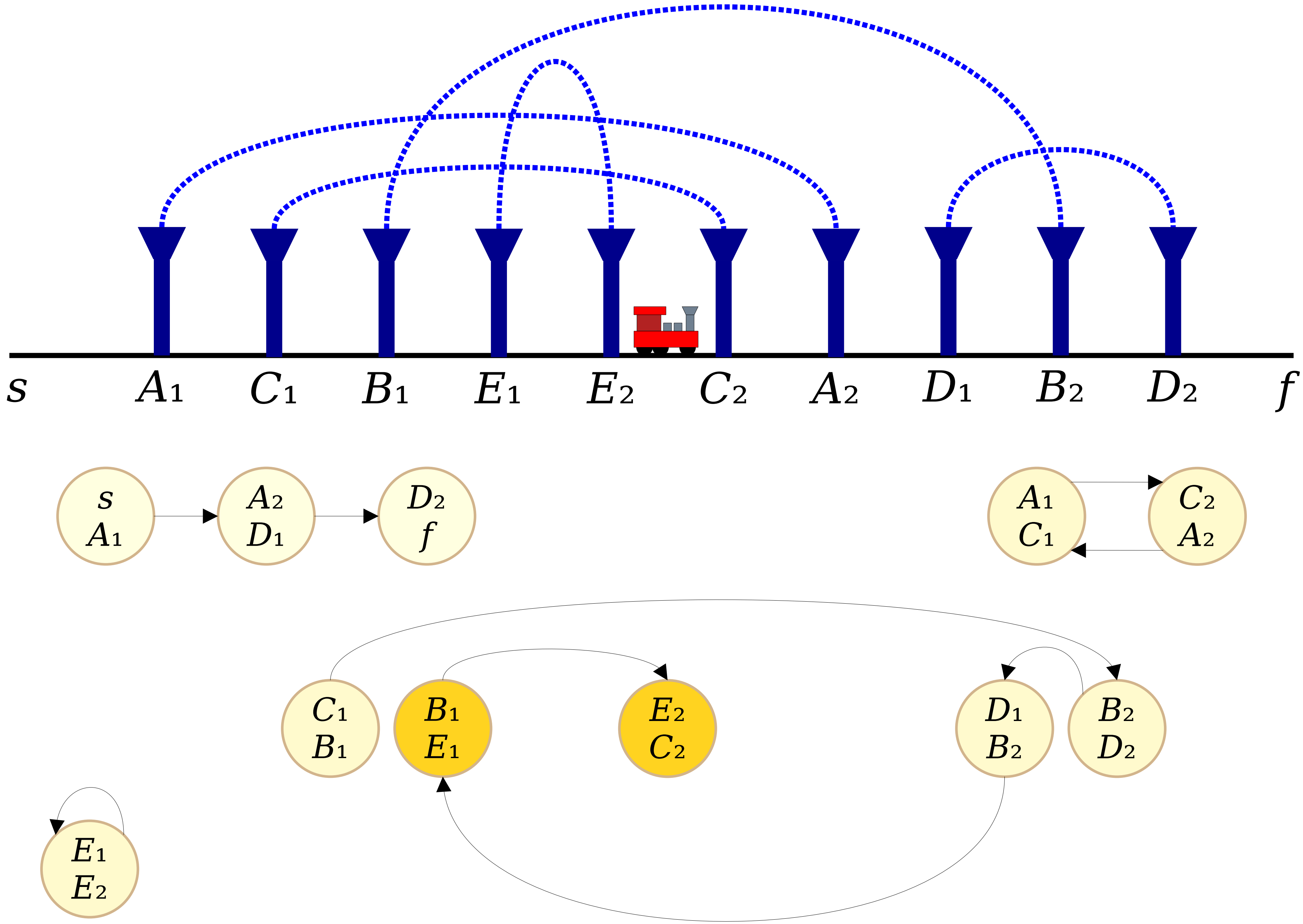

s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_2 A_2
C_1 B_1
B_1 E_1
E_2 C_2
D_1 B_2
B_2 D_2
E_1 E_2



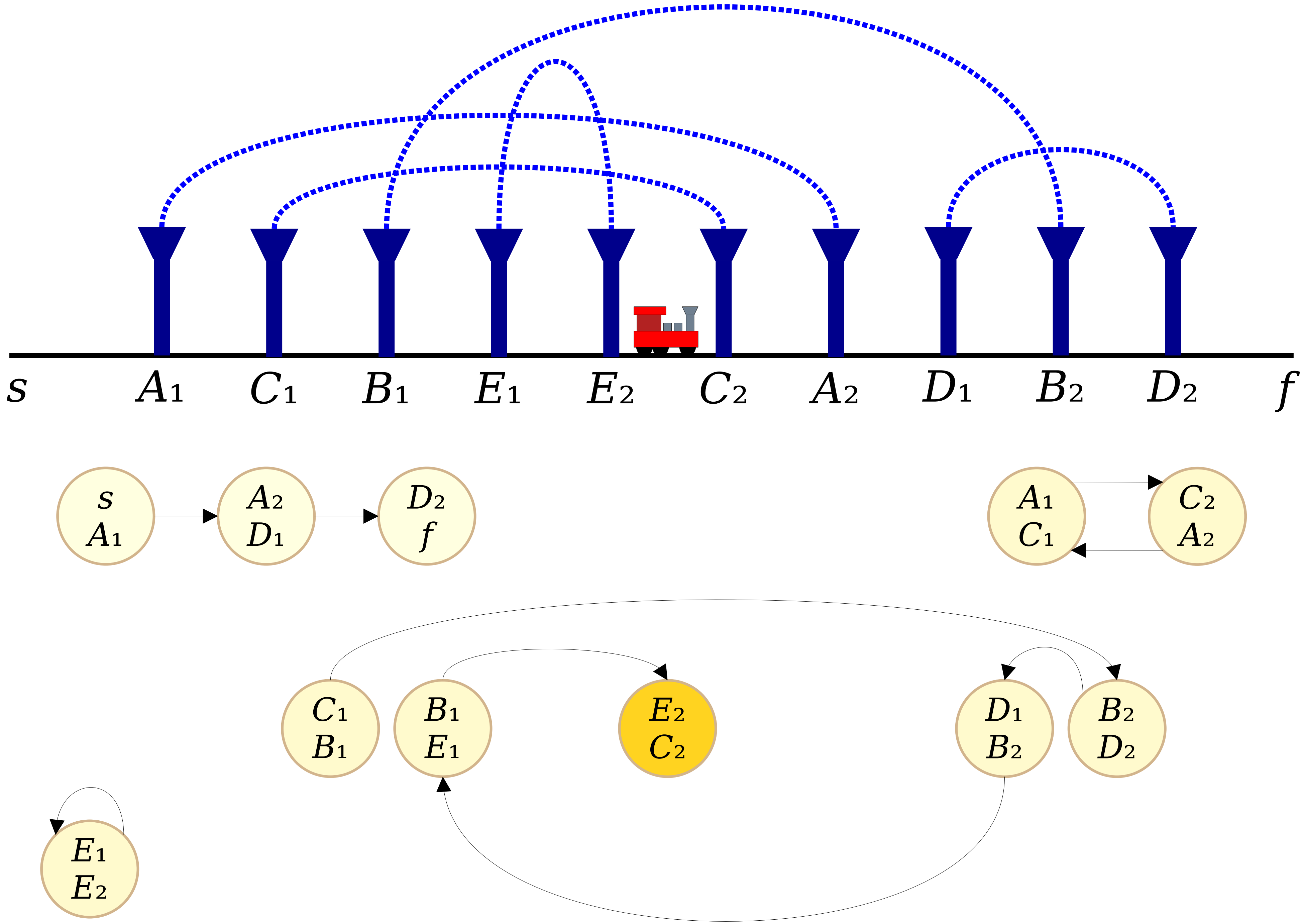
s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B1 E1
E2 C2
D1 B2
B2 D2
E1 E2

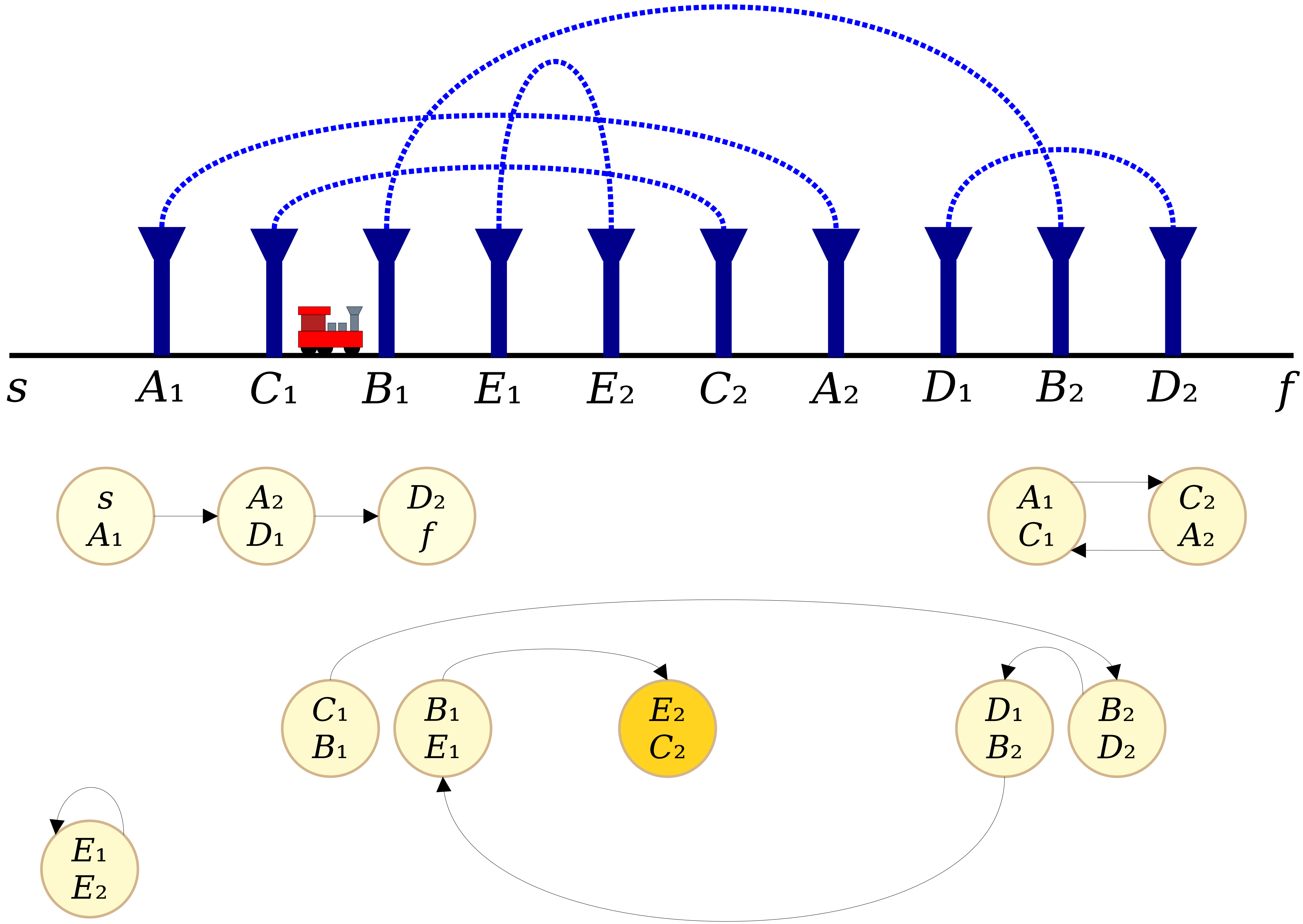


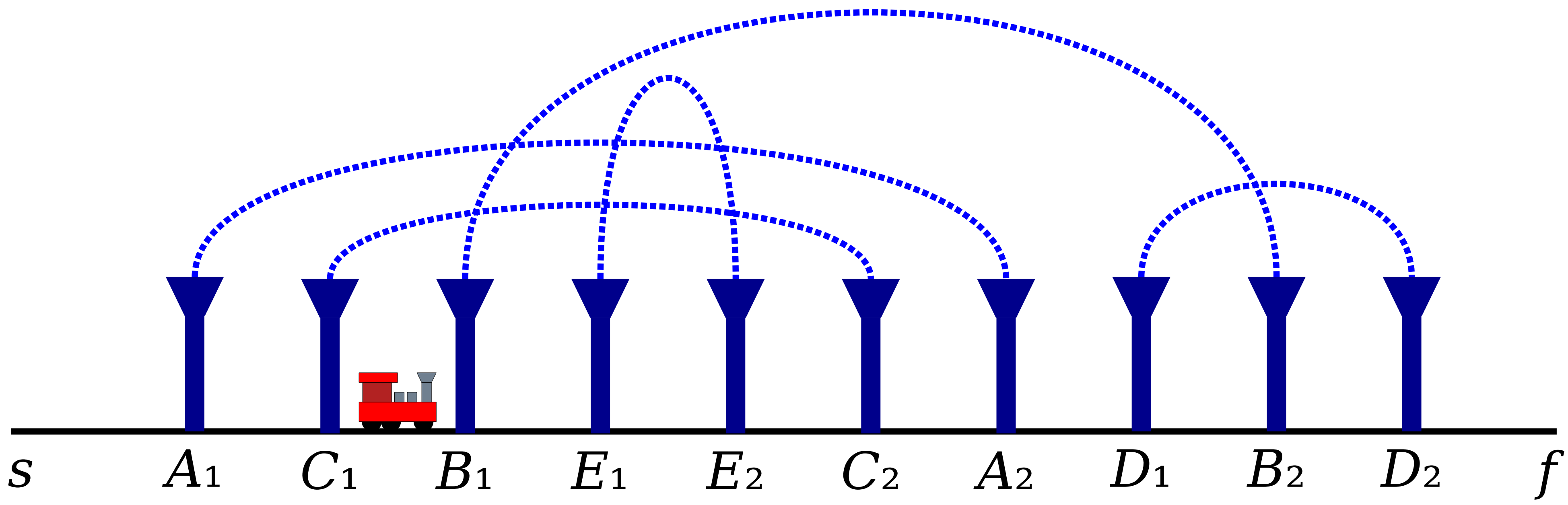
s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B1 E1
E2 C2
D1 B2
B2 D2
E1 E2



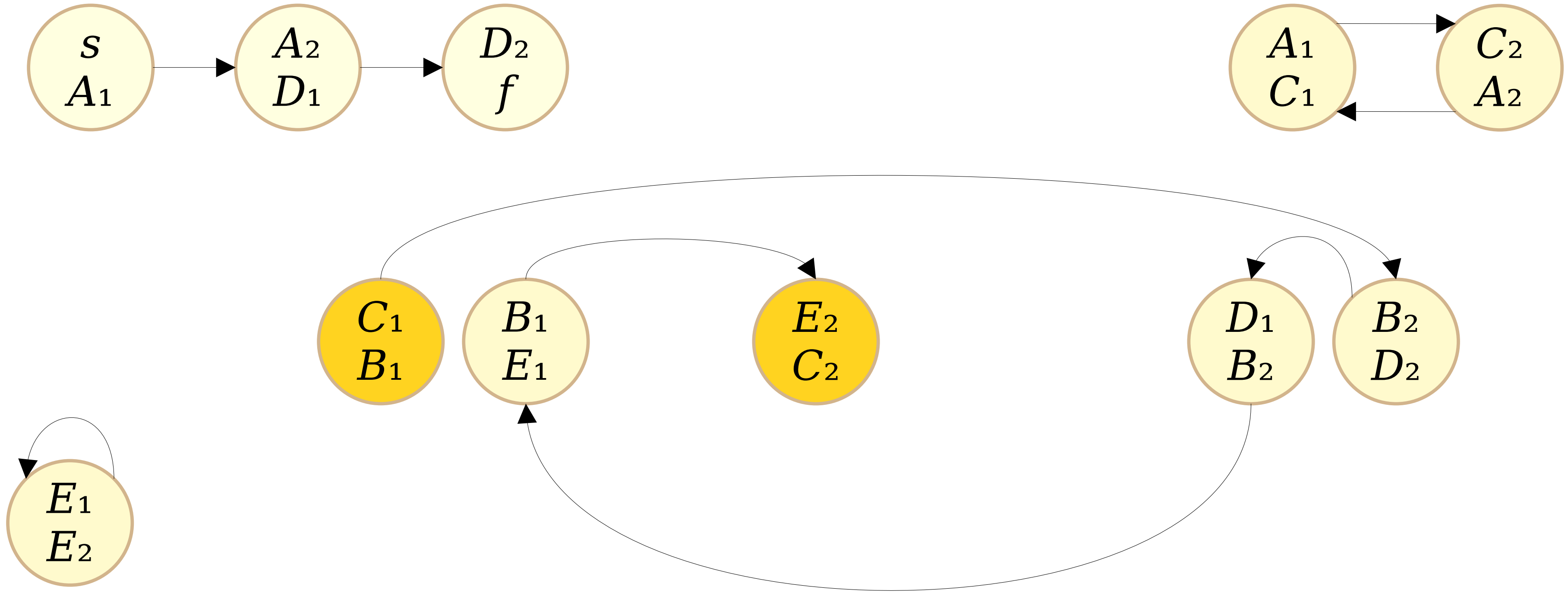
s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_2 A_2
C_1 B_1
B_1 E_1
E_2 C_2
D_1 B_2
B_2 D_2
E_1 E_2


s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_2 A_2
C_1 B_1
B_1 E_1
E_2 C_2
D_1 B_2
B_2 D_2
E_1 E_2

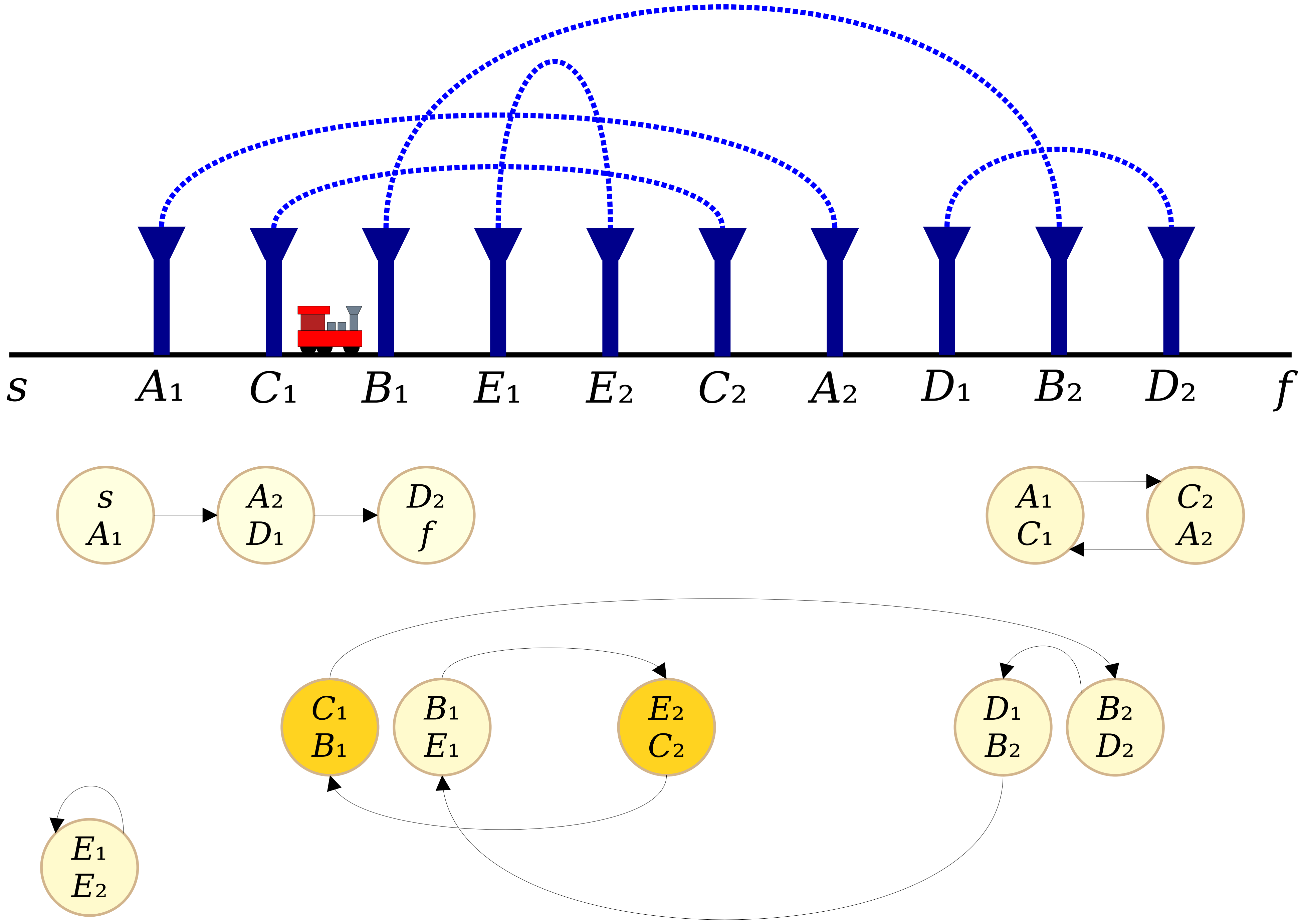

s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B1 E1
E2 C2
D1 B2
B2 D2
E1 E2

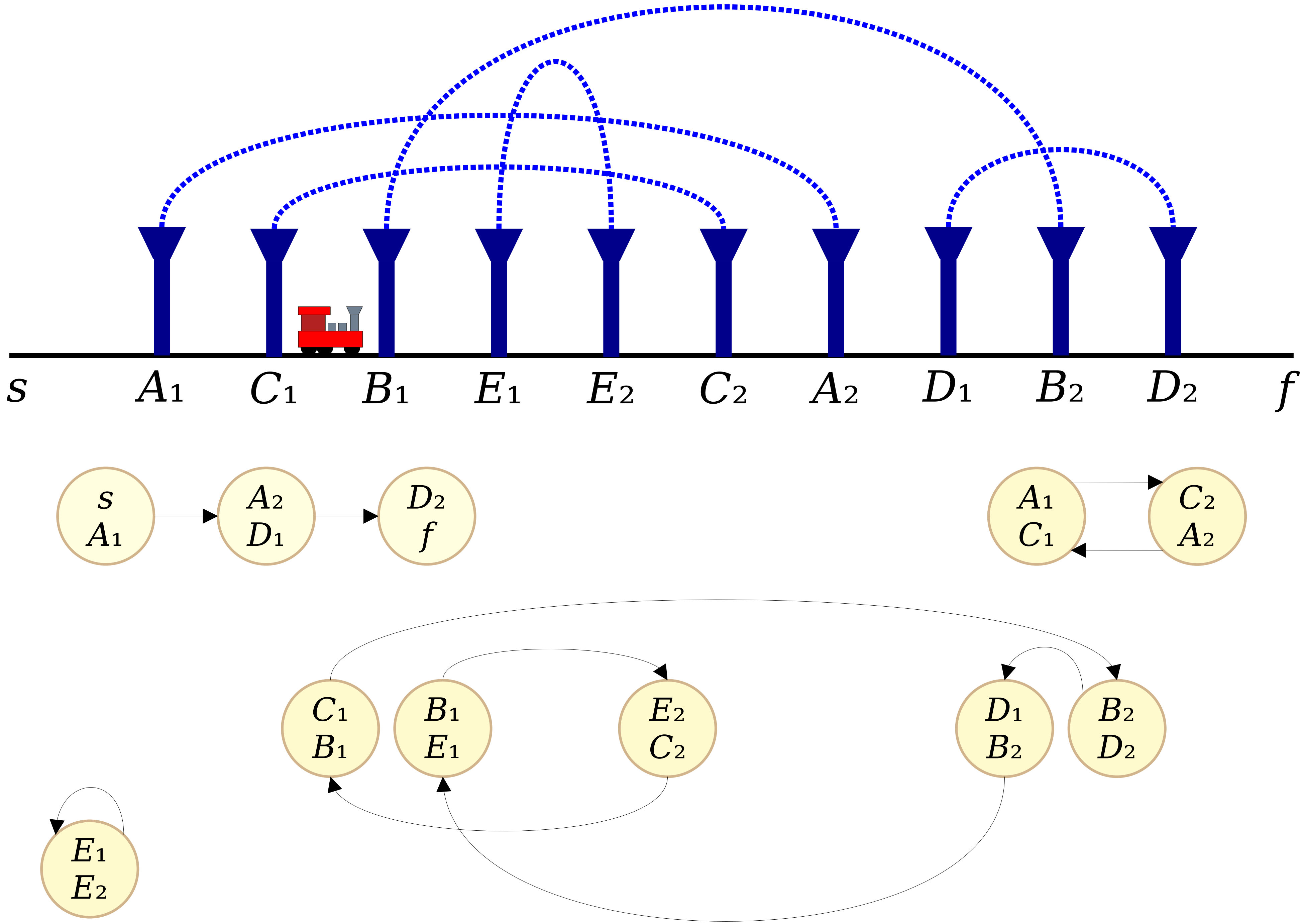


s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f

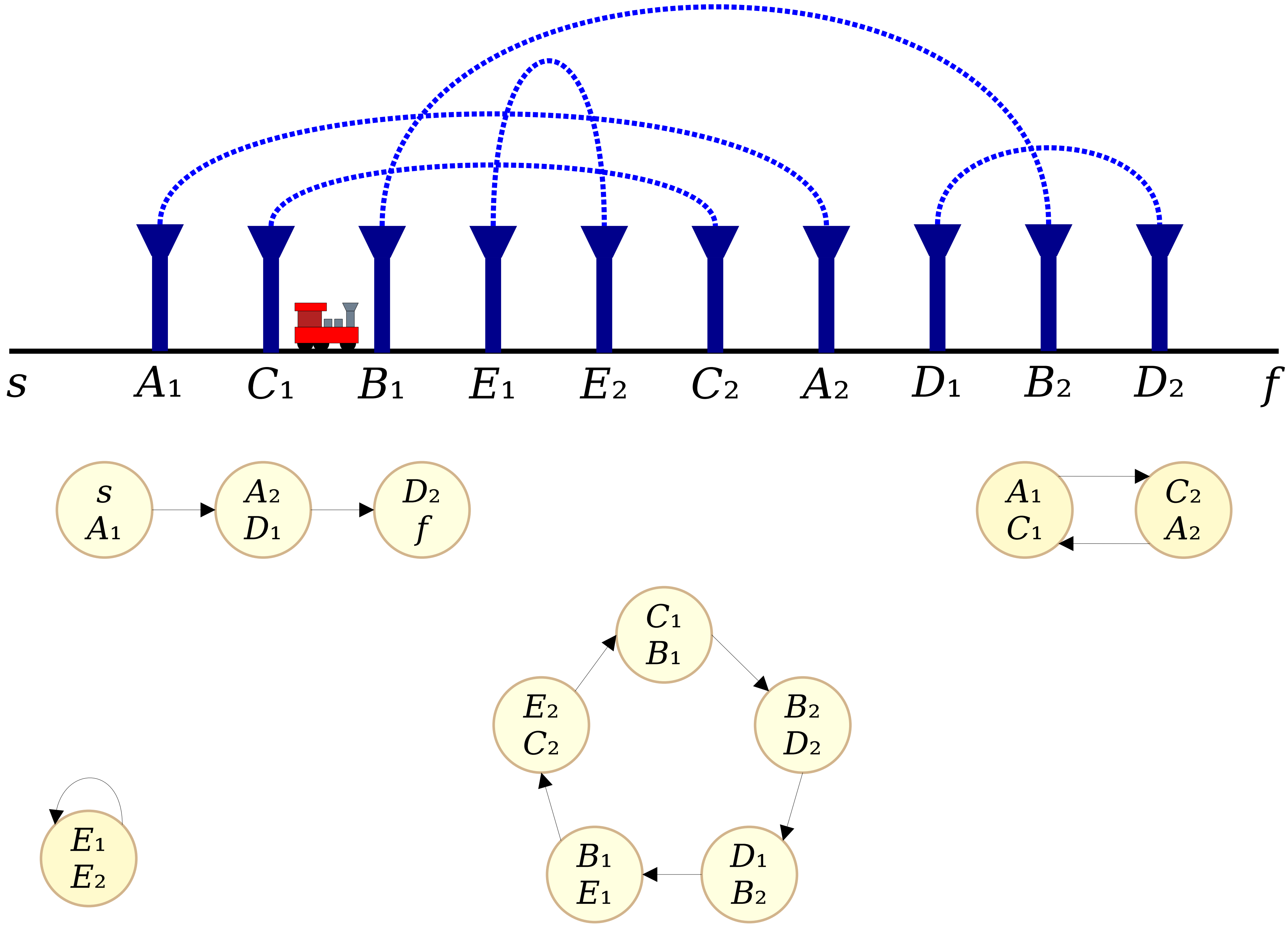


s A_1
A_2 D_1
D_2 f
A_1 C_1
C_2 A_2
C_1 B_1
B_1 E_1
E_2 C_2
D_1 B_2
B_2 D_2
E_1 E_2


s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_2 A_2
C_1 B_1
B_1 E_1
E_2 C_2
D_1 B_2
B_2 D_2
E_1 E_2



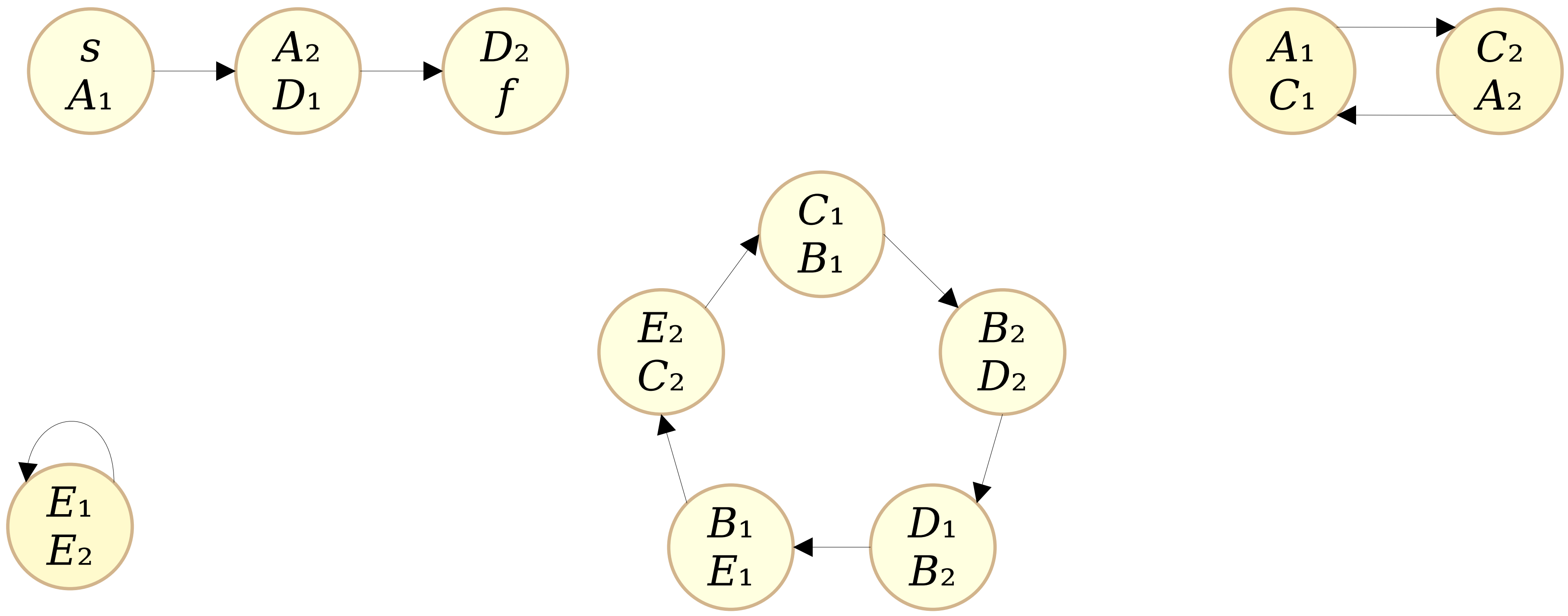
s
A_1
C_1
B_1
E_1
E_2
C_2
A_2
D_1
B_2
D_2
f
s A_1
A_2 D_1
D_2 f
A_1 C_1
C_2 A_2
C_1 B_1
B_1 E_1
E_2 C_2
D_1 B_2
B_2 D_2
E_1 E_2



s
A1
C1
B1
E1
E2
C2
A2
D1
B2
D2
f
s A1
A2 D1
D2 f
A1 C1
C2 A2
C1 B1
B2 D2
D1 B2
B1 E1
E2 C2
E1 E2

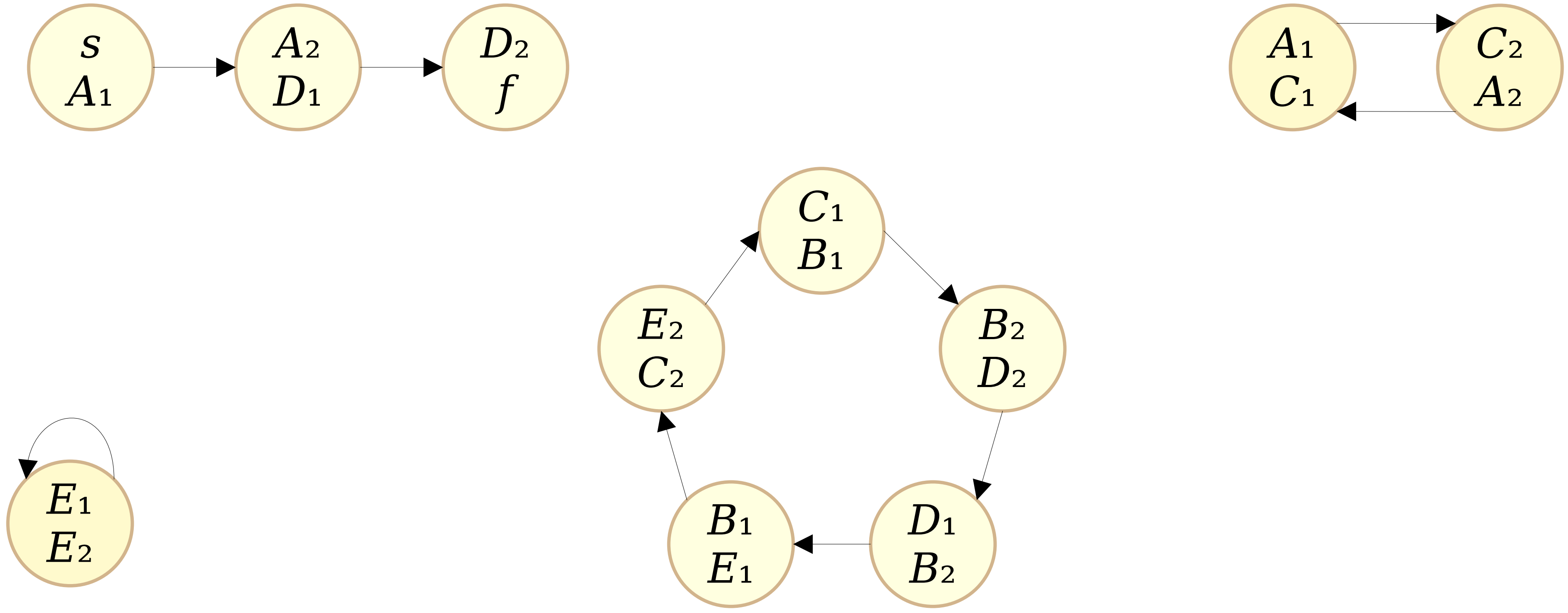
The Teleporter Digraph

- Each line of teleporters gives rise to a directed graph.
  - Each node in the graph represents a segment.
  - Each edge represents following a teleporter.
- That digraph consists of paths and cycles.
- ***Question:*** Why does the digraph look like this?



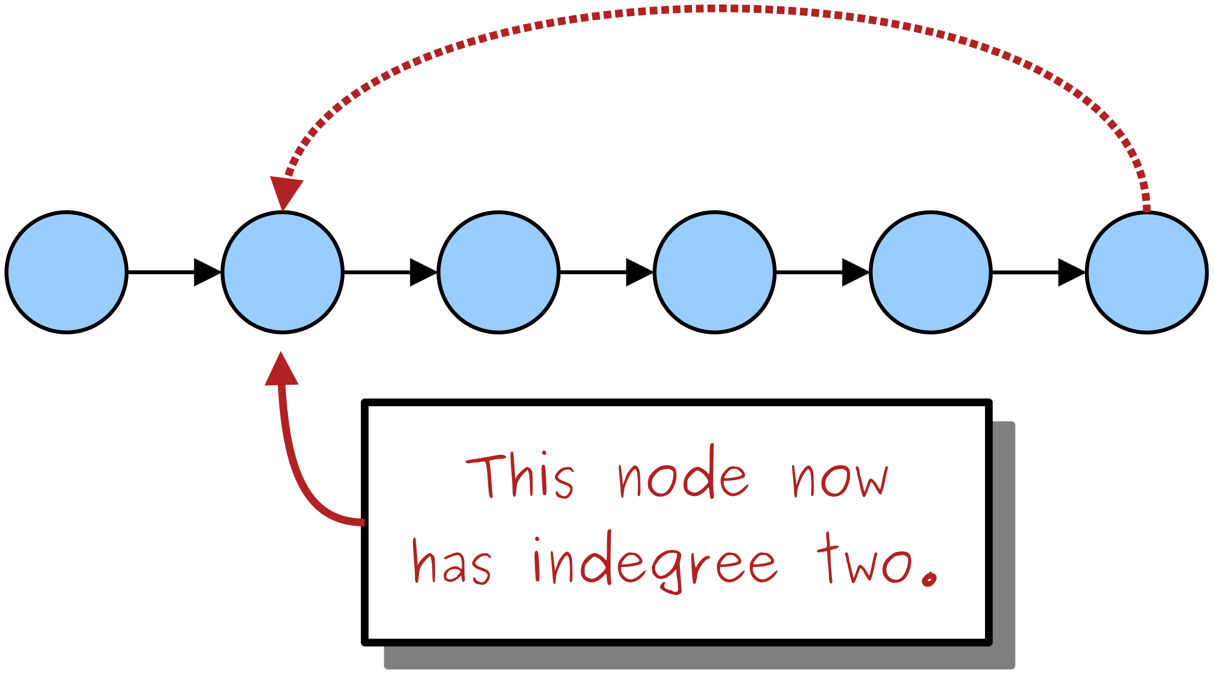
The Teleporter Digraph

- In a directed graph, the ***indegree*** of a node is the number of edges entering that node. The ***outdegree*** of a node is the number of edges leaving that node.
- Notice anything about the indegrees and outdegrees of this digraph?



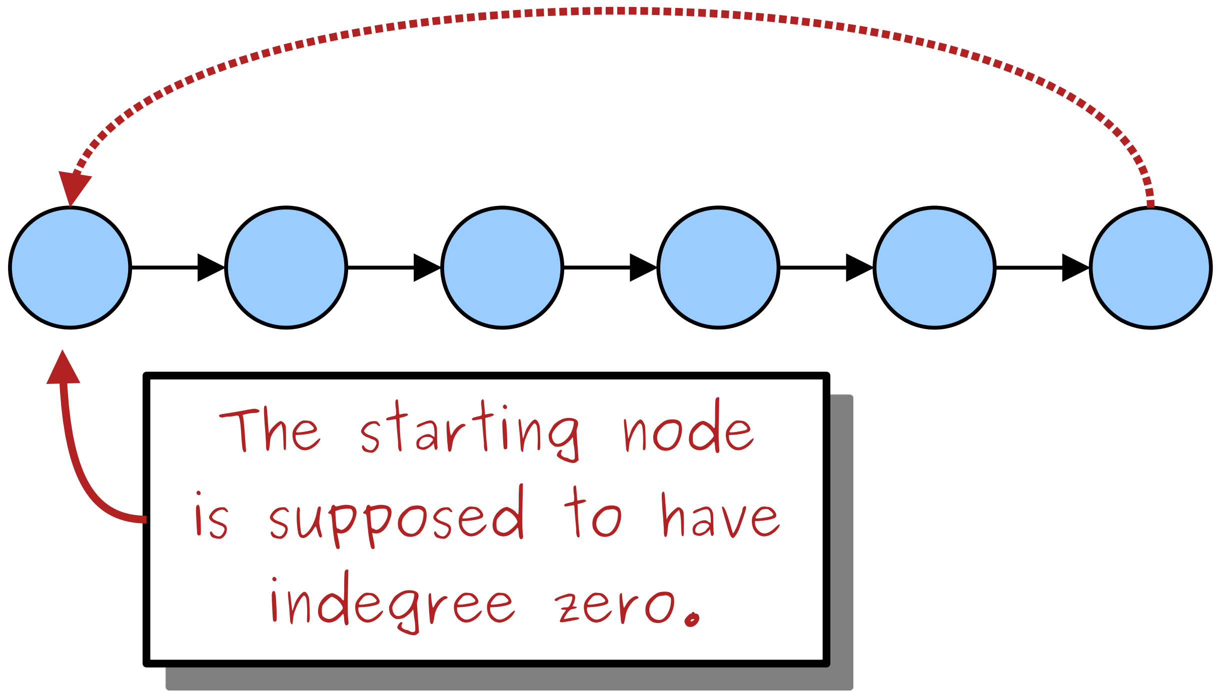
The Teleporter Digraph

- Let $G = (V, E)$ be a digraph where each node's indegree is at most one and each node's outdegree is at most one.
- ***Theorem:*** Any walk starting at a node of indegree zero is also a path.



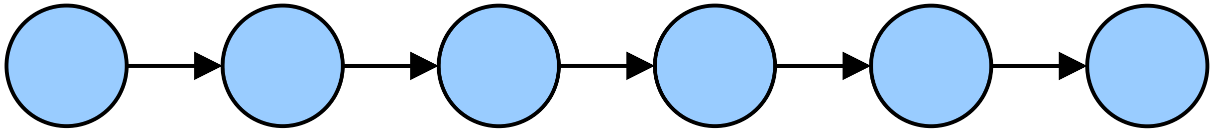
The Teleporter Digraph

- Let $G = (V, E)$ be a digraph where each node's indegree is at most one and each node's outdegree is at most one.
- ***Theorem:*** Any walk starting at a node of indegree zero is also a path.



The Teleporter Digraph

- Let $G = (V, E)$ be a digraph where each node's indegree is at most one and each node's outdegree is at most one.
- ***Theorem:*** Any walk starting at a node of indegree zero is also a path.



Theorem: Let $G = (V, E)$ be a directed graph where each node has indegree at most one and outdegree at most one. Consider any walk T beginning at a node v_0 of indegree zero. Then T is a path.

Theorem: Let $G = (V, E)$ be a directed graph where each node has indegree at most one and outdegree at most one. Consider any walk T beginning at a node v_0 of indegree zero. Then T is a path.

Proof:

Theorem: Let $G = (V, E)$ be a directed graph where each node has indegree at most one and outdegree at most one. Consider any walk T beginning at a node v_0 of indegree zero. Then T is a path.

Proof: Suppose for the sake of contradiction that T is not a path, meaning that it contains a repeated node.

Theorem: Let $G = (V, E)$ be a directed graph where each node has indegree at most one and outdegree at most one. Consider any walk T beginning at a node v_0 of indegree zero. Then T is a path.

Proof: Suppose for the sake of contradiction that T is not a path, meaning that it contains a repeated node. List the nodes in T, stopping just before we list the first repeated node.

Theorem: Let $G = (V, E)$ be a directed graph where each node has indegree at most one and outdegree at most one. Consider any walk T beginning at a node v_0 of indegree zero. Then T is a path.

Proof: Suppose for the sake of contradiction that T is not a path, meaning that it contains a repeated node. List the nodes in T, stopping just before we list the first repeated node. Label the nodes found this way as

$$v_0, v_1, v_2, v_3, \ldots, v_k.$$

Theorem: Let $G = (V, E)$ be a directed graph where each node has indegree at most one and outdegree at most one. Consider any walk T beginning at a node v_0 of indegree zero. Then T is a path.

Proof: Suppose for the sake of contradiction that T is not a path, meaning that it contains a repeated node. List the nodes in T, stopping just before we list the first repeated node. Label the nodes found this way as

$$v_0, v_1, v_2, v_3, \ldots, v_k.$$

Nodes v_0, v_1, …, and v_k are distinct because we've stopped just before revisiting a node.

Theorem: Let $G = (V, E)$ be a directed graph where each node has indegree at most one and outdegree at most one. Consider any walk T beginning at a node v_0 of indegree zero. Then T is a path.

Proof: Suppose for the sake of contradiction that T is not a path, meaning that it contains a repeated node. List the nodes in T, stopping just before we list the first repeated node. Label the nodes found this way as

$$v_0, v_1, v_2, v_3, \ldots, v_k.$$

Nodes v_0, v_1, …, and v_k are distinct because we've stopped just before revisiting a node. We also know that the next node in the walk (call it r) is a repeated node, with (v_k, r) being a directed edge in E.

Theorem: Let $G = (V, E)$ be a directed graph where each node has indegree at most one and outdegree at most one. Consider any walk T beginning at a node v_0 of indegree zero. Then T is a path.

Proof: Suppose for the sake of contradiction that T is not a path, meaning that it contains a repeated node. List the nodes in T, stopping just before we list the first repeated node. Label the nodes found this way as

$$v_0, v_1, v_2, v_3, \ldots, v_k.$$

Nodes v_0, v_1, …, and v_k are distinct because we've stopped just before revisiting a node. We also know that the next node in the walk (call it r) is a repeated node, with (v_k, r) being a directed edge in E. We now ask: which earlier node is r equal to?

Theorem: Let $G = (V, E)$ be a directed graph where each node has indegree at most one and outdegree at most one. Consider any walk T beginning at a node v_0 of indegree zero. Then T is a path.

Proof: Suppose for the sake of contradiction that T is not a path, meaning that it contains a repeated node. List the nodes in T, stopping just before we list the first repeated node. Label the nodes found this way as

$$v_0, v_1, v_2, v_3, \ldots, v_k.$$

Nodes $v_0, v_1, \ldots,$ and v_k are distinct because we've stopped just before revisiting a node. We also know that the next node in the walk (call it r) is a repeated node, with (v_k, r) being a directed edge in E. We now ask: which earlier node is r equal to?

Case 1: $r = v_0$.

Case 2: $r = v_i$ for some $i \neq 0$.

Theorem: Let $G = (V, E)$ be a directed graph where each node has indegree at most one and outdegree at most one. Consider any walk T beginning at a node v_0 of indegree zero. Then T is a path.

Proof: Suppose for the sake of contradiction that T is not a path, meaning that it contains a repeated node. List the nodes in T, stopping just before we list the first repeated node. Label the nodes found this way as

$$v_0, v_1, v_2, v_3, \ldots, v_k.$$

Nodes v_0, v_1, …, and v_k are distinct because we've stopped just before revisiting a node. We also know that the next node in the walk (call it r) is a repeated node, with (v_k, r) being a directed edge in E. We now ask: which earlier node is r equal to?

Case 1: $r = v_0$. This means that (v_k, v_0) is a directed edge, which is impossible because v_0 has indegree zero.

Case 2*:* $r = v_i$ for some $i \neq 0$.

Theorem: Let $G = (V, E)$ be a directed graph where each node has indegree at most one and outdegree at most one. Consider any walk T beginning at a node v_0 of indegree zero. Then T is a path.

Proof: Suppose for the sake of contradiction that T is not a path, meaning that it contains a repeated node. List the nodes in T, stopping just before we list the first repeated node. Label the nodes found this way as

$$v_0, v_1, v_2, v_3, \ldots, v_k.$$

Nodes $v_0, v_1, \ldots,$ and v_k are distinct because we've stopped just before revisiting a node. We also know that the next node in the walk (call it r) is a repeated node, with (v_k, r) being a directed edge in E. We now ask: which earlier node is r equal to?

Case 1: $r = v_0$. This means that (v_k, v_0) is a directed edge, which is impossible because v_0 has indegree zero.

Case 2: $r = v_i$ for some $i \neq 0$. Then (v_{i-1}, v_i) and (v_k, v_i) are directed edges in G, which is impossible because v_i has indegree one.

Theorem: Let $G = (V, E)$ be a directed graph where each node has indegree at most one and outdegree at most one. Consider any walk T beginning at a node v_0 of indegree zero. Then T is a path.

Proof: Suppose for the sake of contradiction that T is not a path, meaning that it contains a repeated node. List the nodes in T, stopping just before we list the first repeated node. Label the nodes found this way as

$$v_0, v_1, v_2, v_3, \ldots, v_k.$$

Nodes $v_0, v_1, \ldots,$ and v_k are distinct because we've stopped just before revisiting a node. We also know that the next node in the walk (call it r) is a repeated node, with (v_k, r) being a directed edge in E. We now ask: which earlier node is r equal to?

Case 1: $r = v_0$. This means that (v_k, v_0) is a directed edge, which is impossible because v_0 has indegree zero.

Case 2: $r = v_i$ for some $i \neq 0$. Then (v_{i-1}, v_i) and (v_k, v_i) are directed edges in G, which is impossible because v_i has indegree one.

In either case we've reached a contradiction, so our assumption must have been wrong.

Theorem: Let $G = (V, E)$ be a directed graph where each node has indegree at most one and outdegree at most one. Consider any walk T beginning at a node v_0 of indegree zero. Then T is a path.

Proof: Suppose for the sake of contradiction that T is not a path, meaning that it contains a repeated node. List the nodes in T, stopping just before we list the first repeated node. Label the nodes found this way as

$$v_0, v_1, v_2, v_3, \ldots, v_k.$$

Nodes v_0, v_1, …, and v_k are distinct because we've stopped just before revisiting a node. We also know that the next node in the walk (call it r) is a repeated node, with (v_k, r) being a directed edge in E. We now ask: which earlier node is r equal to?

Case 1: $r = v_0$. This means that (v_k, v_0) is a directed edge, which is impossible because v_0 has indegree zero.

Case 2: $r = v_i$ for some $i \neq 0$. Then (v_{i-1}, v_i) and (v_k, v_i) are directed edges in G, which is impossible because v_i has indegree one.

In either case we've reached a contradiction, so our assumption must have been wrong. Thus T is a path.

Theorem: Let $G = (V, E)$ be a directed graph where each node has indegree at most one and outdegree at most one. Consider any walk T beginning at a node v_0 of indegree zero. Then T is a path.

Proof: Suppose for the sake of contradiction that T is not a path, meaning that it contains a repeated node. List the nodes in T, stopping just before we list the first repeated node. Label the nodes found this way as

$$v_0, v_1, v_2, v_3, \ldots, v_k.$$

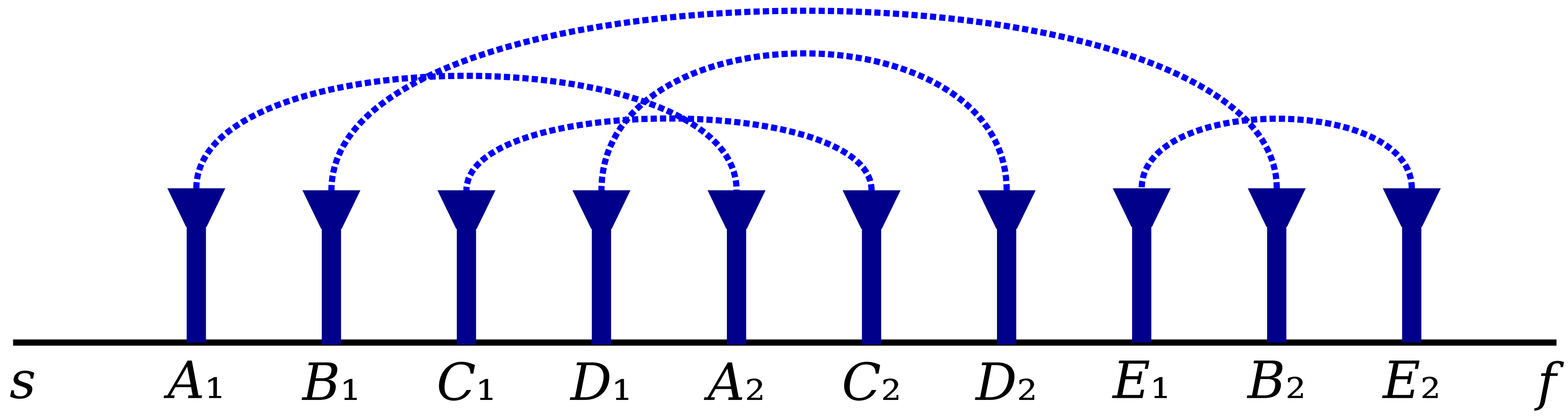
Nodes $v_0, v_1, \ldots,$ and v_k are distinct because we've stopped just before revisiting a node. We also know that the next node in the walk (call it r) is a repeated node, with (v_k, r) being a directed edge in E. We now ask: which earlier node is r equal to?

Case 1: $r = v_0$. This means that (v_k, v_0) is a directed edge, which is impossible because v_0 has indegree zero.

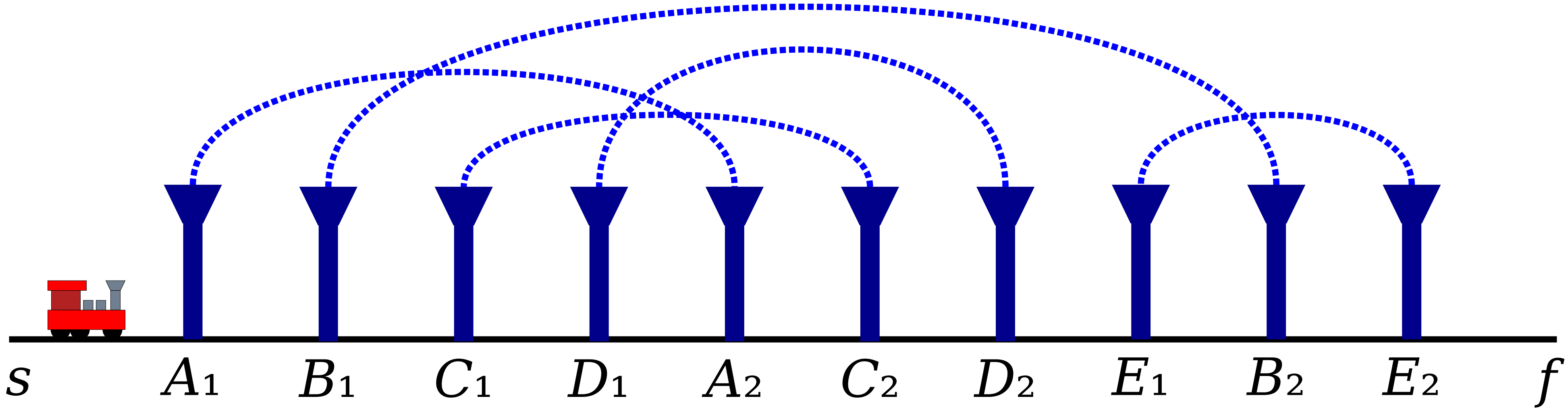
Case 2*:* $r = v_i$ for some $i \neq 0$. Then (v_{i-1}, v_i) and (v_k, v_i) are directed edges in G, which is impossible because v_i has indegree one.

In either case we've reached a contradiction, so our assumption must have been wrong. Thus T is a path. ■

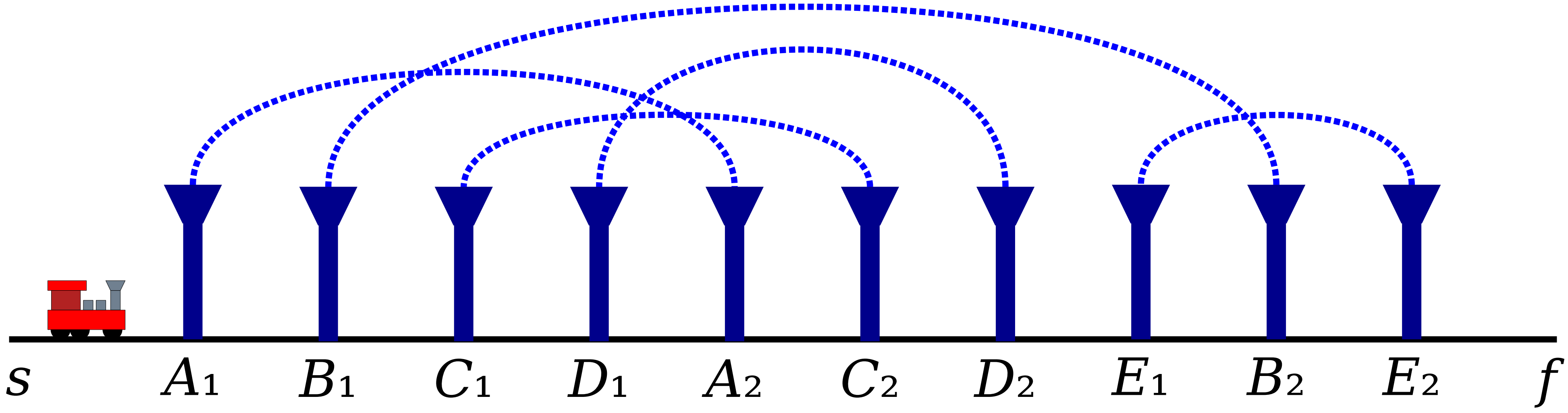
Trapping the Train



Trapping the Train

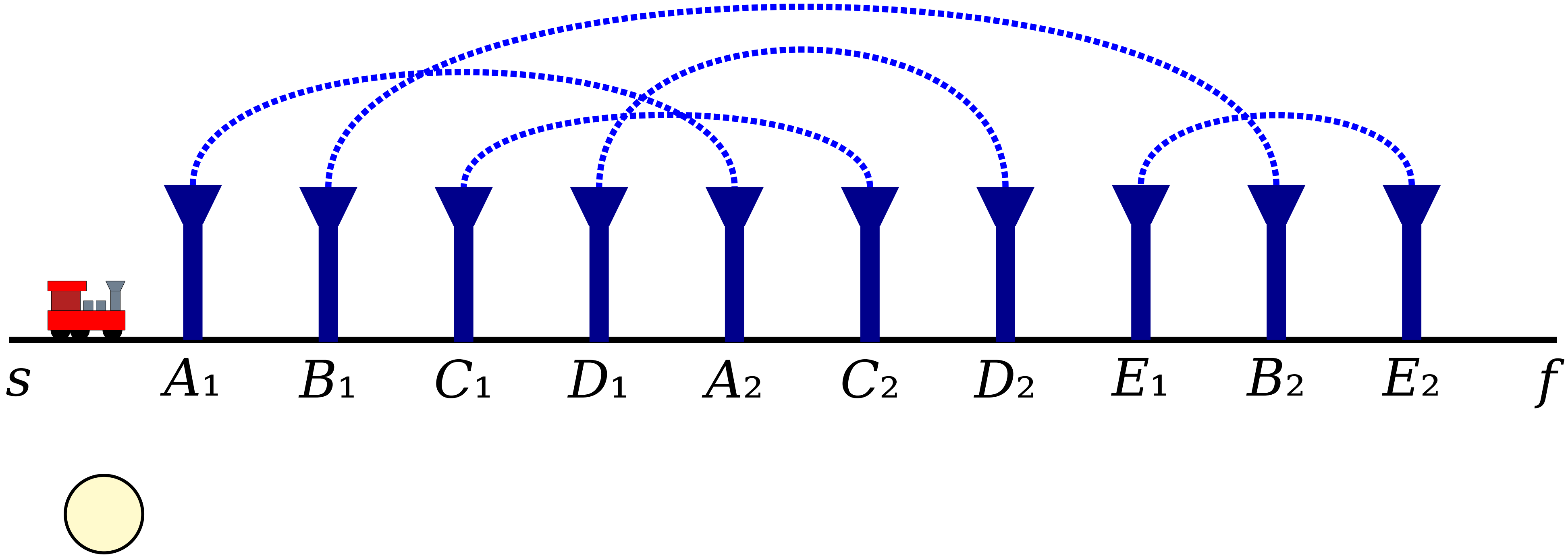


Trapping the Train



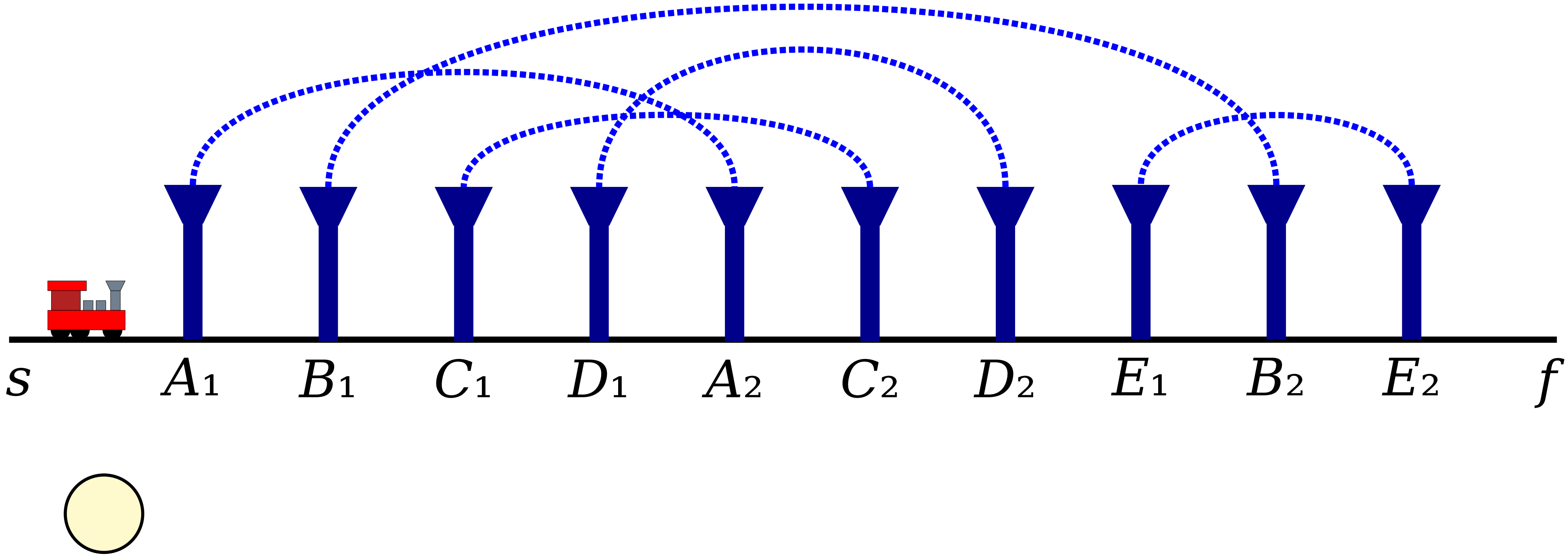
The train begins before the first teleporter, so the start node has indegree zero.

Trapping the Train



The train begins before the first teleporter, so the start node has indegree zero.

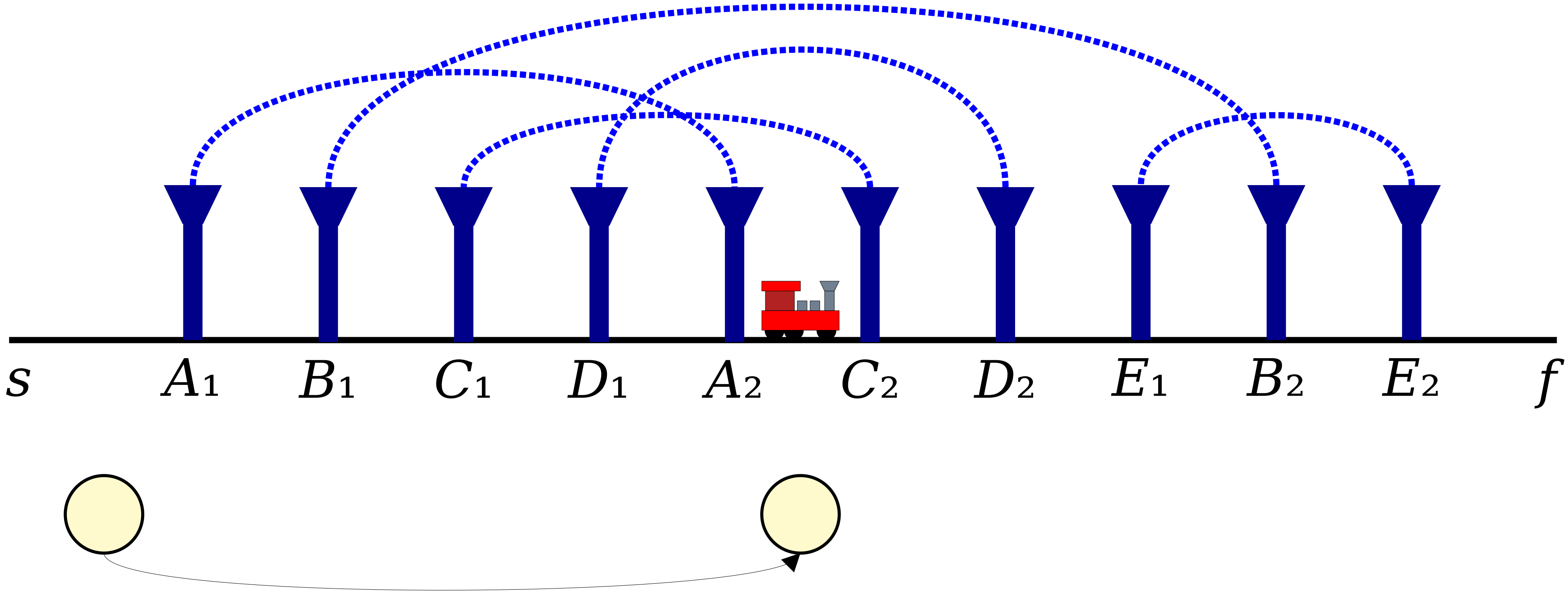
Trapping the Train



The train begins before the first teleporter, so the start node has indegree zero.

Therefore, the walk we trace out is a path, and so it has to end somewhere.

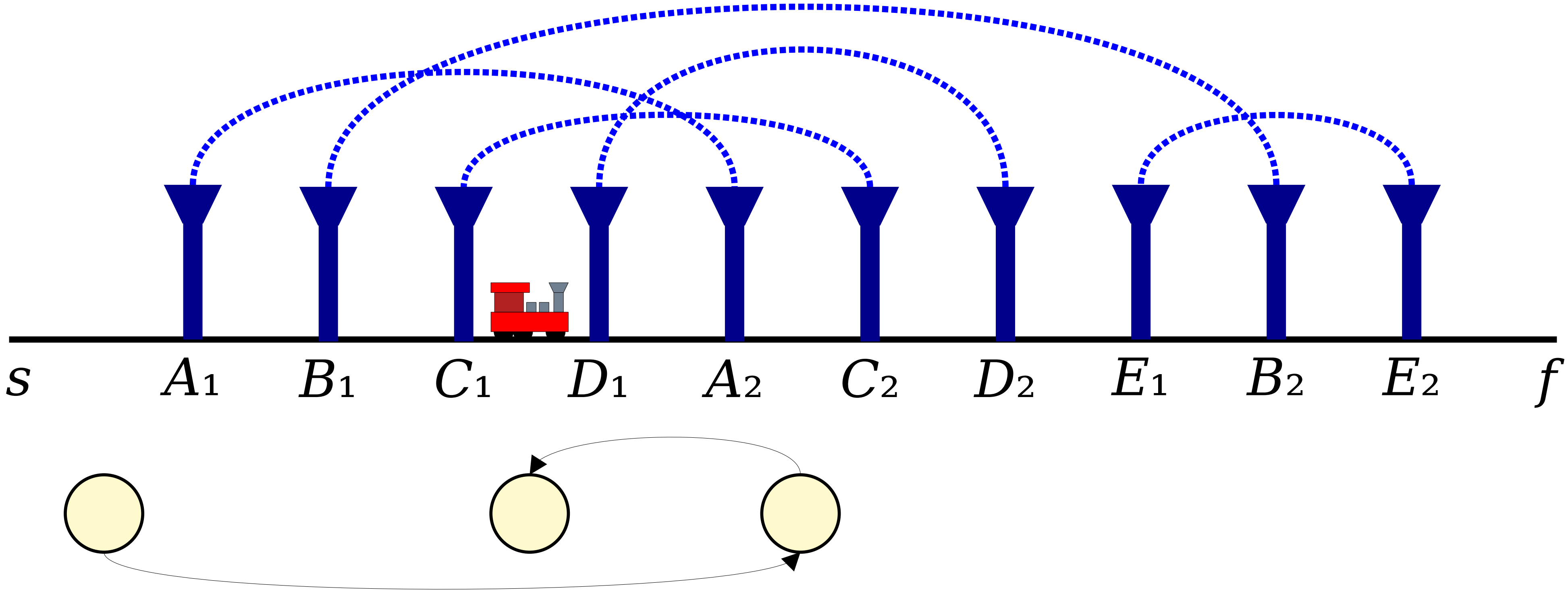
Trapping the Train



The train begins before the first teleporter, so the start node has indegree zero.

Therefore, the walk we trace out is a path, and so it has to end somewhere.

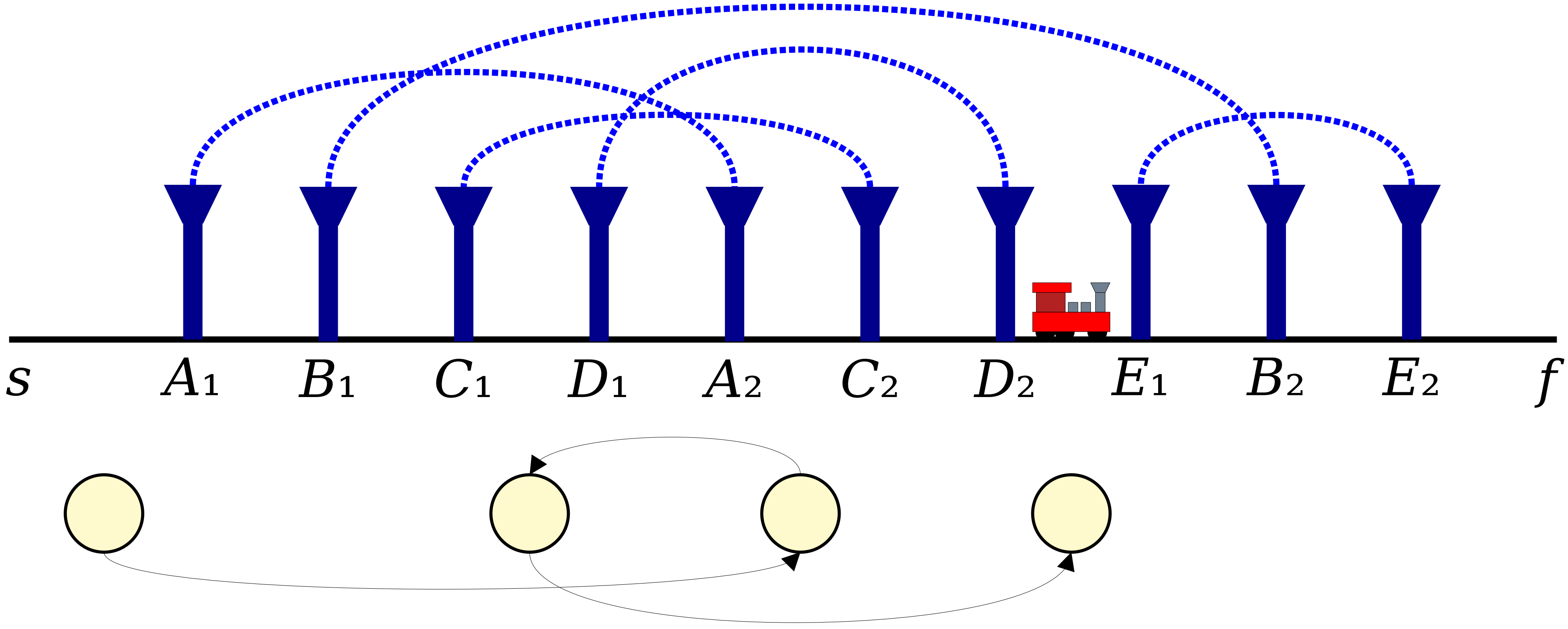
Trapping the Train



The train begins before the first teleporter, so the start node has indegree zero.

Therefore, the walk we trace out is a path, and so it has to end somewhere.

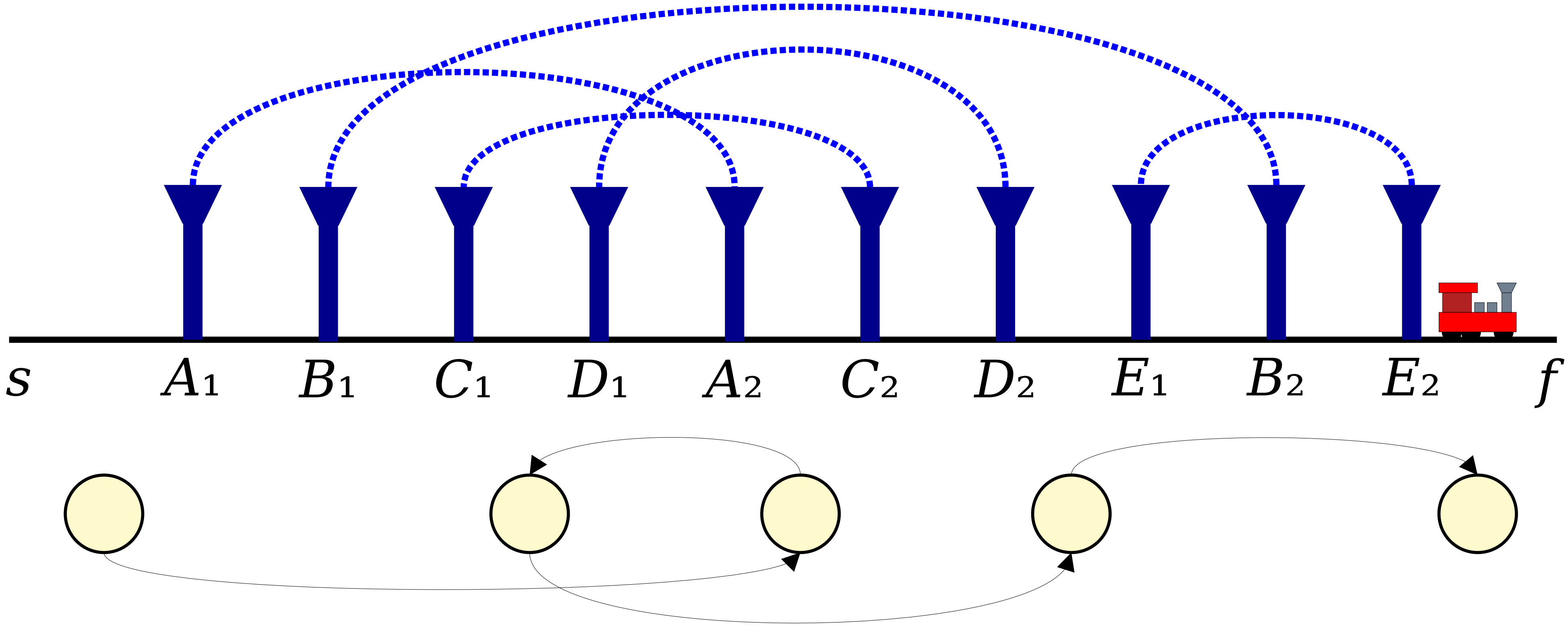
Trapping the Train



The train begins before the first teleporter, so the start node has indegree zero.

Therefore, the walk we trace out is a path, and so it has to end somewhere.

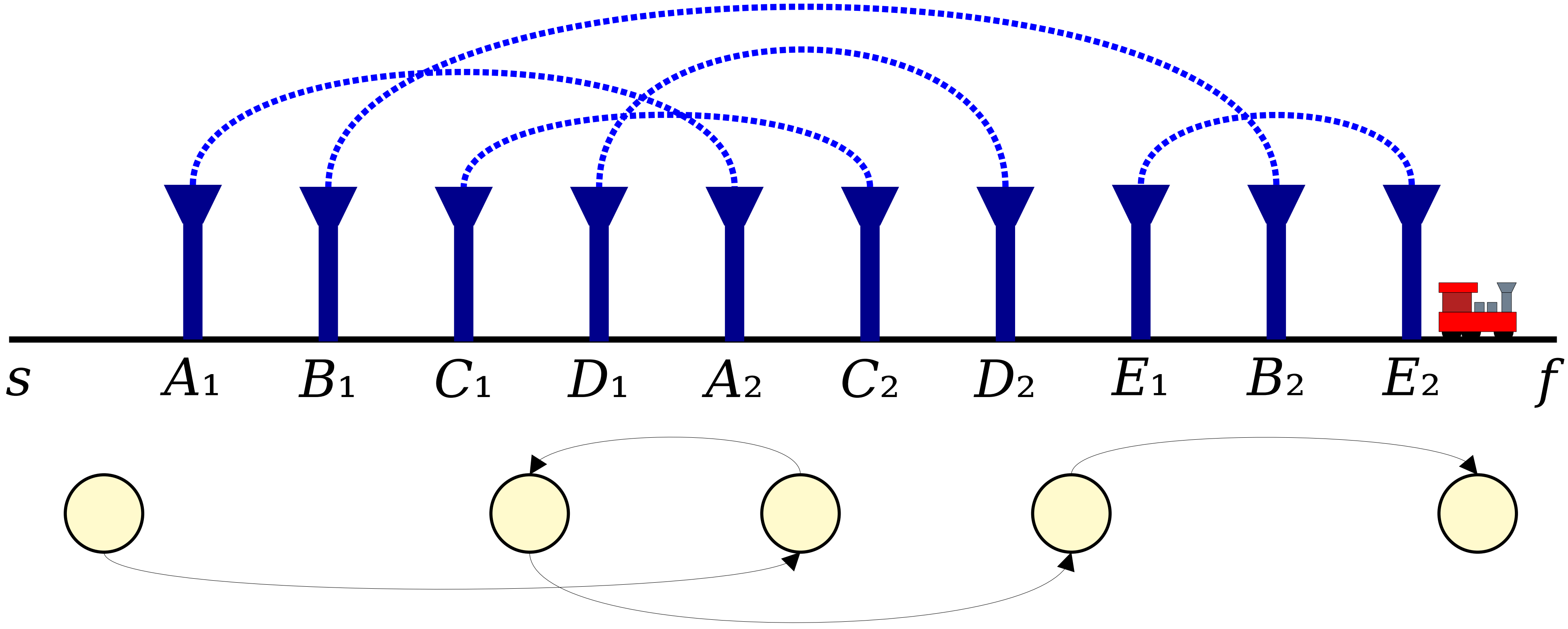
Trapping the Train



The train begins before the first teleporter, so the start node has indegree zero.

Therefore, the walk we trace out is a path, and so it has to end somewhere.

Trapping the Train

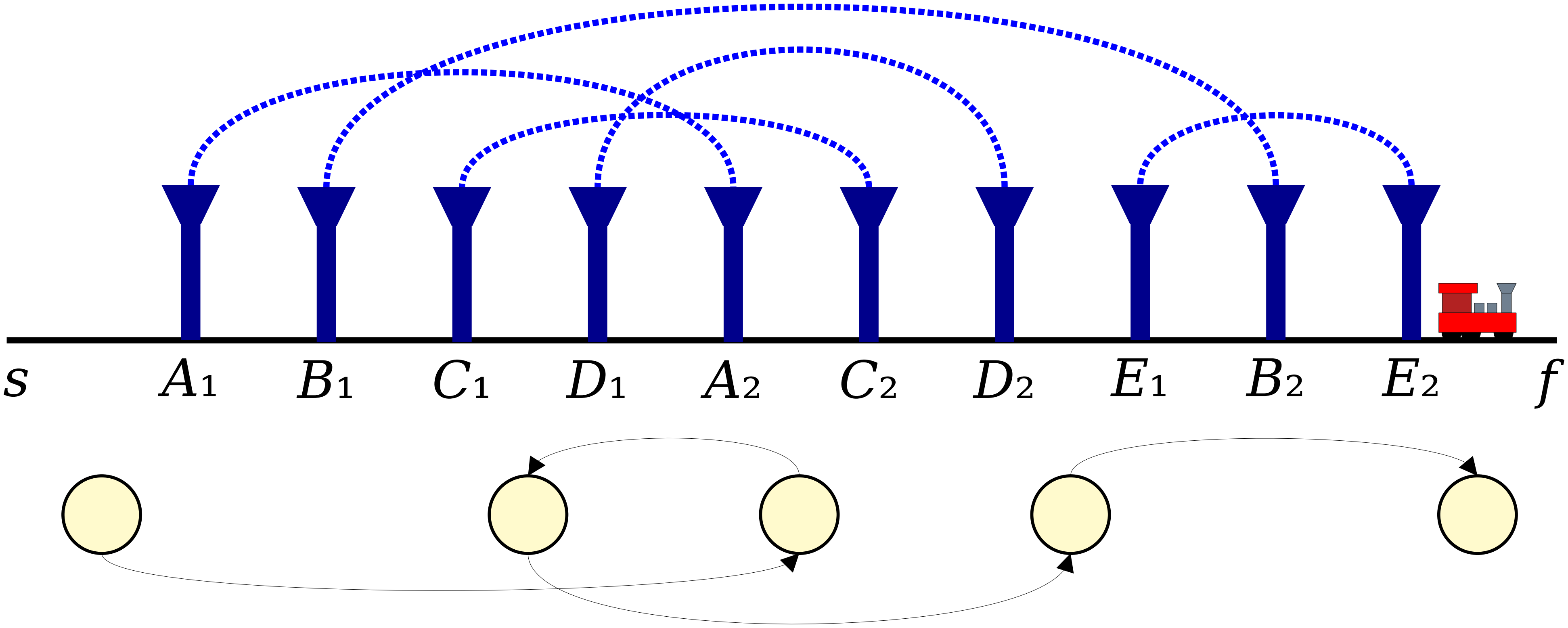


The train begins before the first teleporter, so the start node has indegree zero.

Therefore, the walk we trace out is a path, and so it has to end somewhere.

The only node of outdegree zero is the one after the last teleporter, where the goal is.

Trapping the Train



Theorem: It is impossible to trap the train if it starts before the first teleporter.

Theorem: It is not possible to trap the train in the Teleported Train Problem.

Proof: Consider any arrangement of teleporters. We will prove that the train makes it to the end without getting stuck in a loop.

Divide the train track into segments denoting the ranges between two teleporters or between a teleporter and the start/end of the track. From these segments, construct a directed graph whose nodes are the segments and where there's an edge from a segment S_1 to a segment S_2 if, upon reaching the end of segment S_1, the train teleports to the start of segment S_2.

We claim that every node in this graph has indegree at most one and outdegree at most one. To see this, pick any segment. If that segment begins with a teleporter, then it has one incoming edge that originates at the segment that ends with the paired teleporter. If that segment ends with a teleporter, then it has one outgoing edge to the start of the segment with the paired teleporter.

Now, consider the walk traced out by the train from the starting segment. That segment has indegree zero because it does not begin with a teleporter, so by our previous theorem this walk is a path. There are only finitely many segments and our path never revisits one, so eventually the path ends at a node with outdegree zero. The only node with this property is the end segment, so the train eventually reaches the end of the track. ■

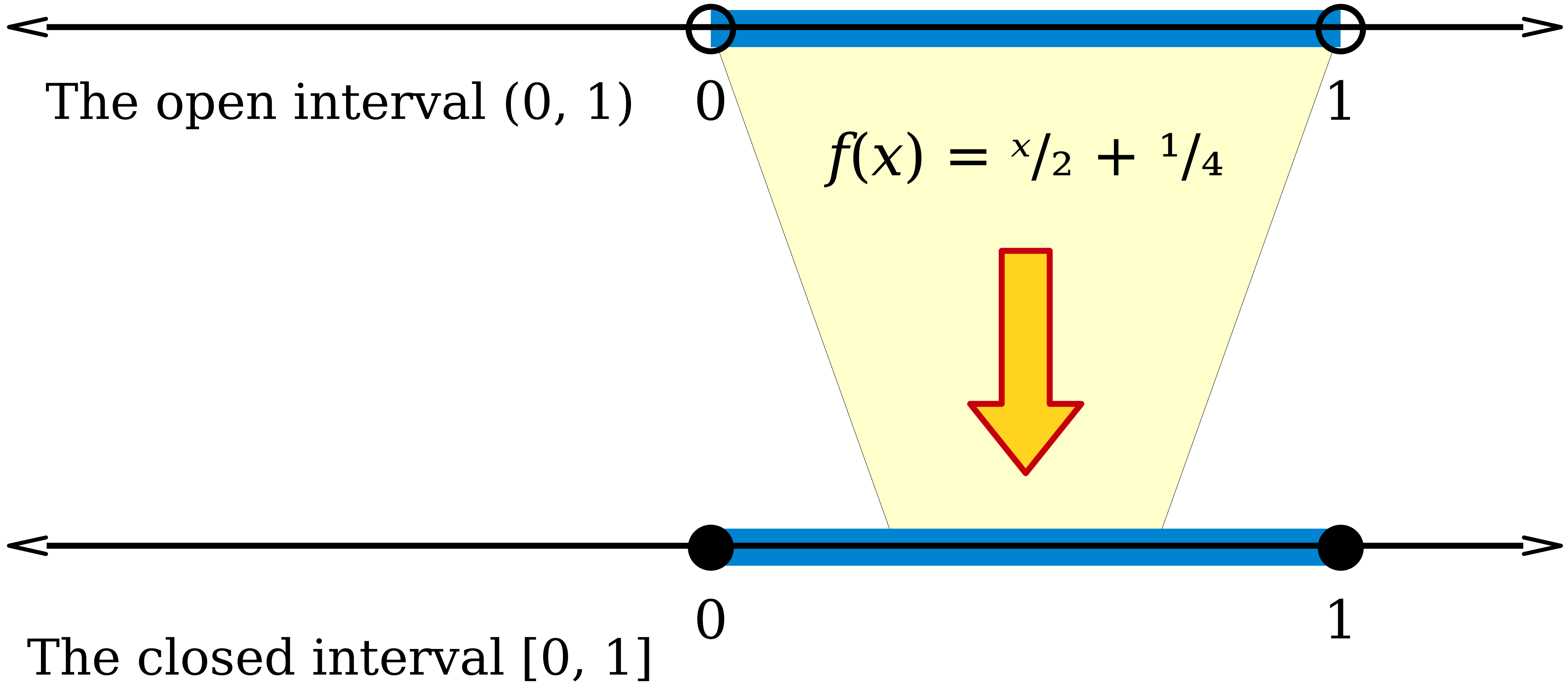
The Cantor-Bernstein-Schroeder Theorem

Theorem (Cantor-Bernstein-Schroeder):

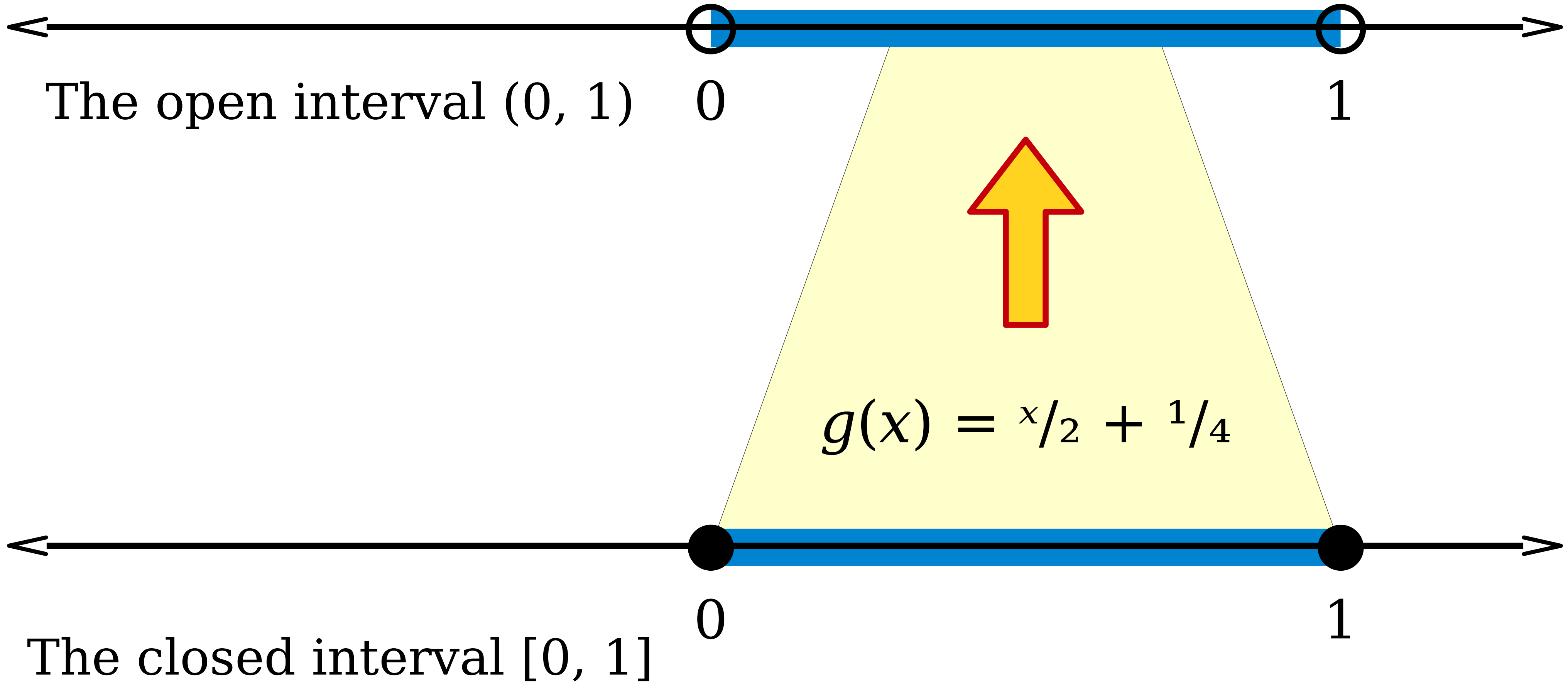
Let S and T be sets. If $|S| \leq |T|$ and $|T| \leq |S|$, then $|S| = |T|$.

(This was first proven by Richard Dedekind.)

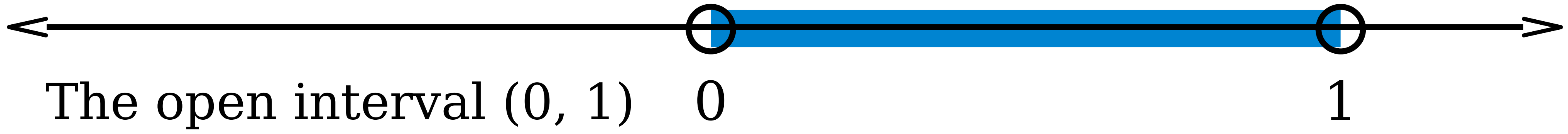
Theorem (Cantor-Bernstein-Schroeder):
Let S and T be sets. If there is an injection $f : S \to T$ and an injection $g : T \to S$, then there is a bijection $h : S \to T$.



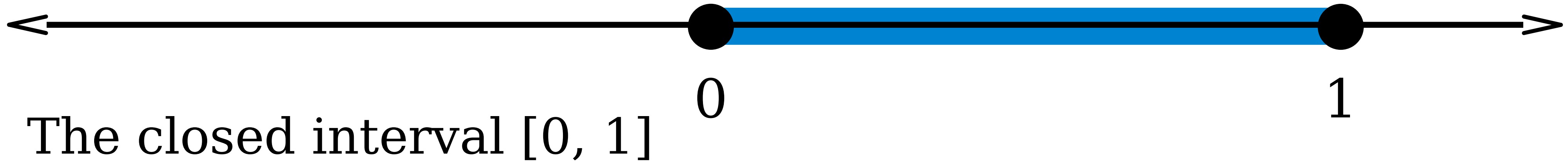
Theorem (Cantor-Bernstein-Schroeder): Let S and T be sets. If there is an injection $f : S \to T$ and an injection $g : T \to S$, then there is a bijection $h : S \to T$.


The open interval (0, 1)
0
1
$g(x) = {}^{x}/_{2} + {}^{1}/_{4}$
0
1
The closed interval [0, 1]

Theorem (Cantor-Bernstein-Schroeder): Let S and T be sets. If there is an injection $f : S \to T$ and an injection $g : T \to S$, then there is a bijection $h : S \to T$.



There's a bijection between these sets – though finding a formula for one is hard enough to be an Optional Fun Problem.



Theorem (Cantor-Bernstein-Schroeder): Let S and T be sets. If there is an injection $f : S \to T$ and an injection $g : T \to S$, then there is a bijection $h : S \to T$.

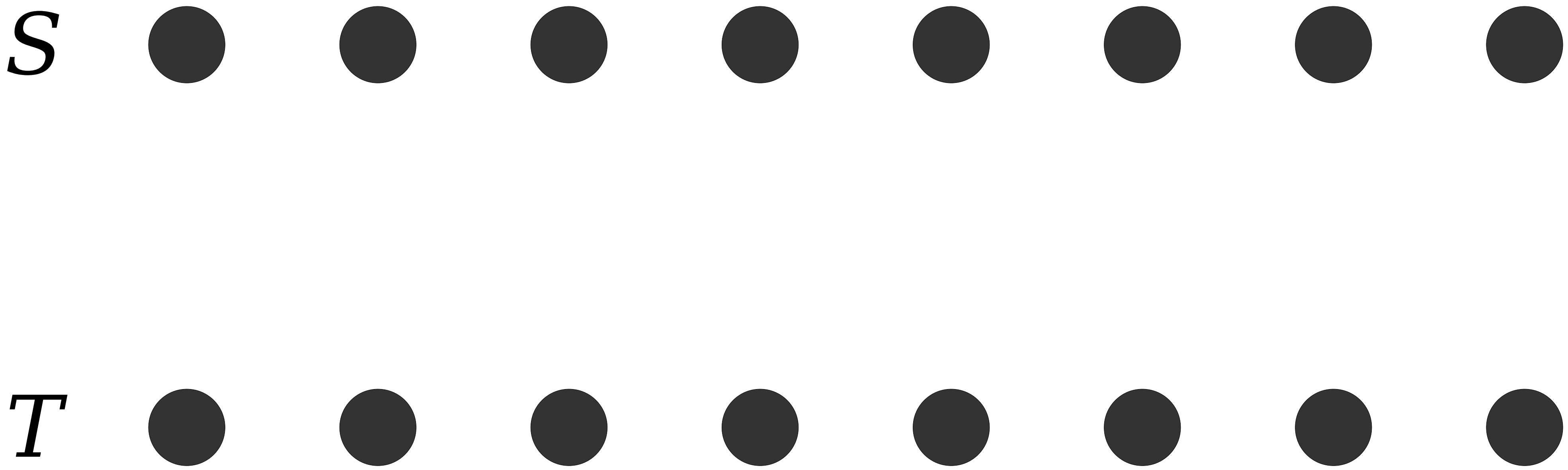
$$f : \mathbb{N} \to \mathbb{N}^2$$
$$g : \mathbb{N}^2 \to \mathbb{N}$$

$$f(n) = (0, n)$$
$$g(m, n) = 2^m \cdot 3^n$$

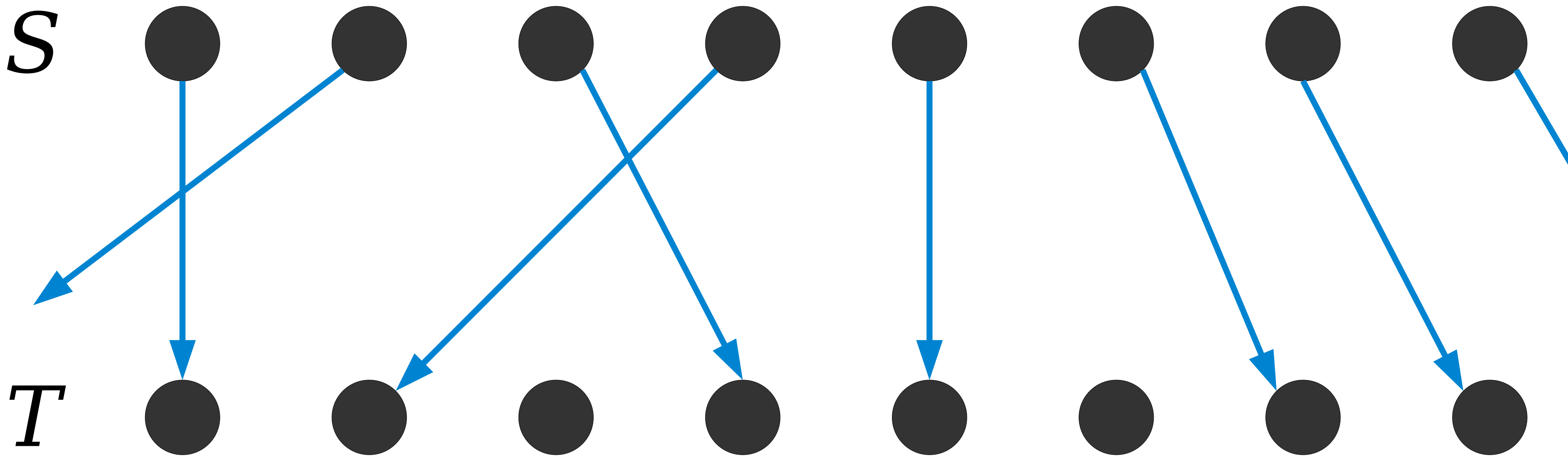
These functions are injective.

Challenge: Find a bijection $h : \mathbb{N} \to \mathbb{N}^2$.

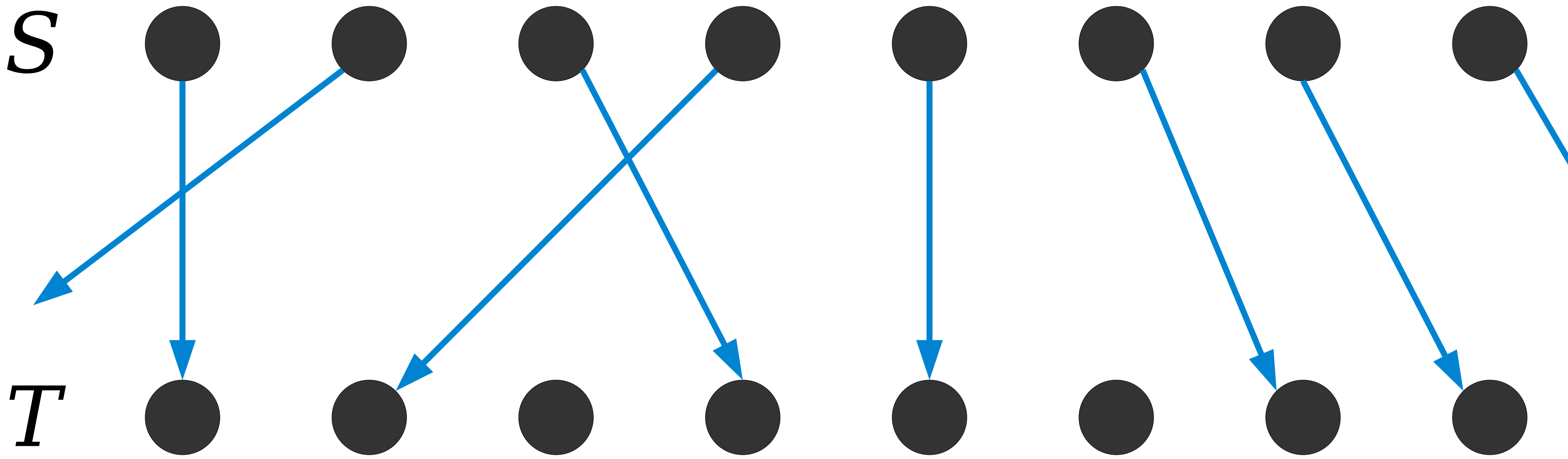
Theorem (Cantor-Bernstein-Schroeder): Let S and T be sets. If there is an injection $f : S \to T$ and an injection $g : T \to S$, then there is a bijection $h : S \to T$.



Theorem (Cantor-Bernstein-Schroeder): Let S and T be sets. If there is an injection $f : S \to T$ and an injection $g : T \to S$, then there is a bijection $h : S \to T$.

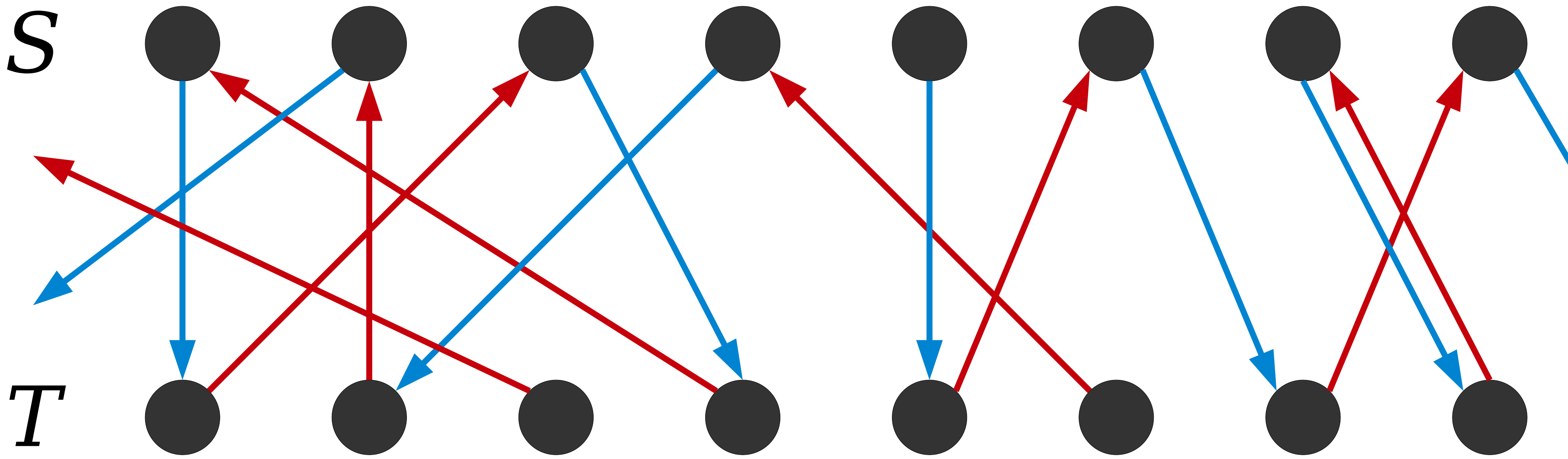


Theorem (Cantor-Bernstein-Schroeder): Let S and T be sets. If there is an injection $f : S \to T$ and an injection $g : T \to S$, then there is a bijection $h : S \to T$.



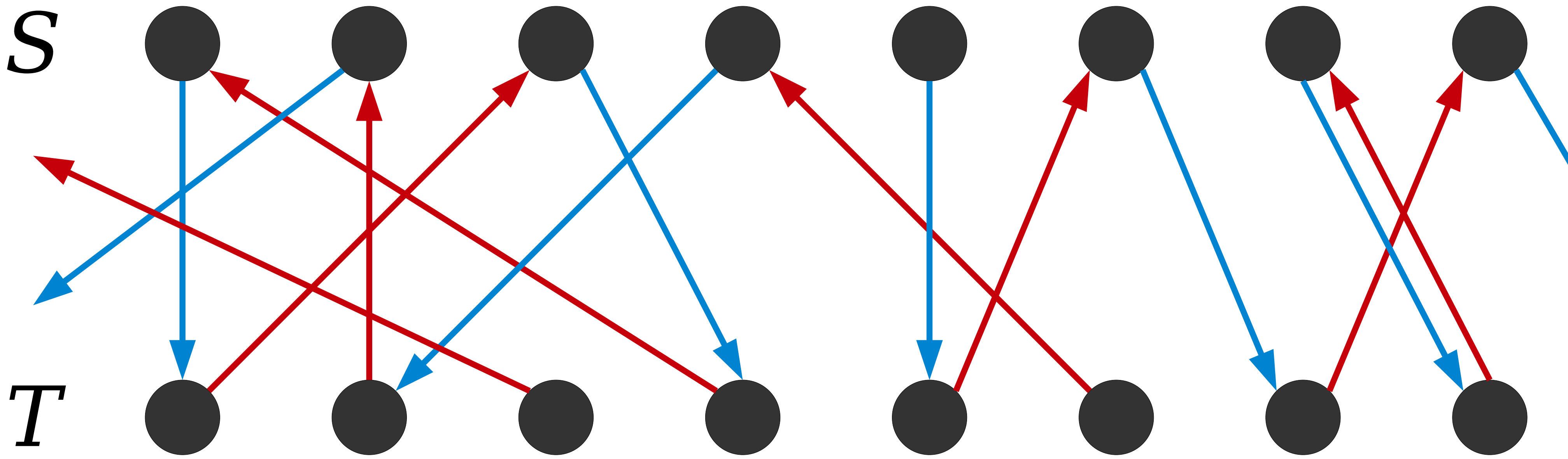
Blue lines represent the injection $f : S \to T$

Theorem (Cantor-Bernstein-Schroeder): Let S and T be sets. If there is an injection $f : S \to T$ and an injection $g : T \to S$, then there is a bijection $h : S \to T$.



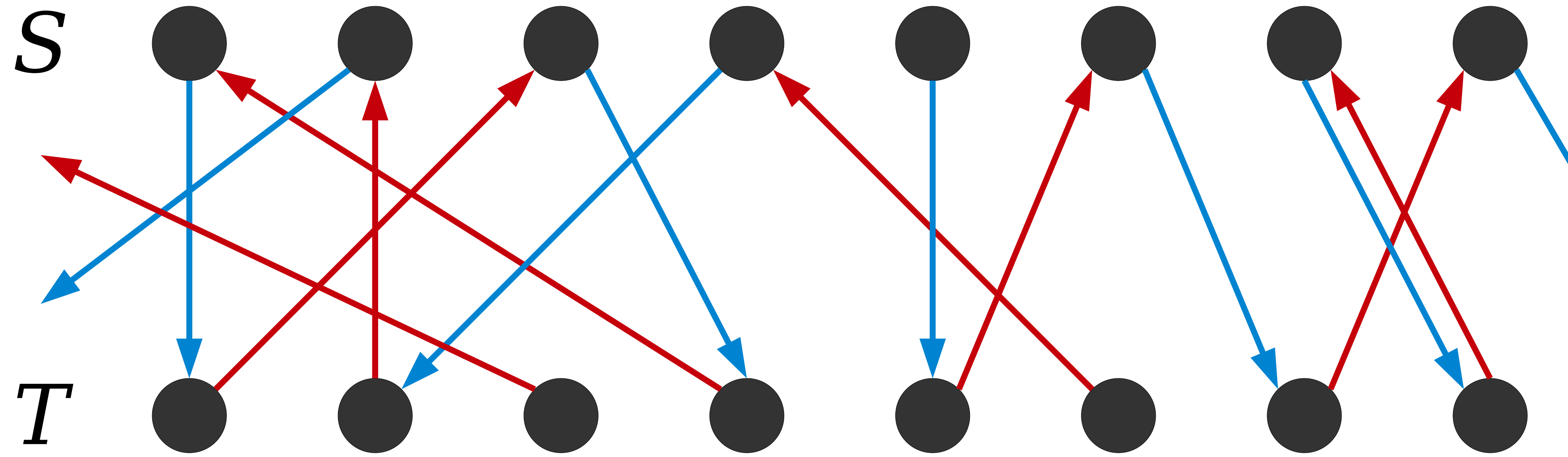
Blue lines represent the injection $f : S \to T$

Theorem (Cantor-Bernstein-Schroeder): Let S and T be sets. If there is an injection $f : S \to T$ and an injection $g : T \to S$, then there is a bijection $h : S \to T$.



Blue lines represent the injection $f : S \to T$
Red lines represent the injection $g : T \to S$

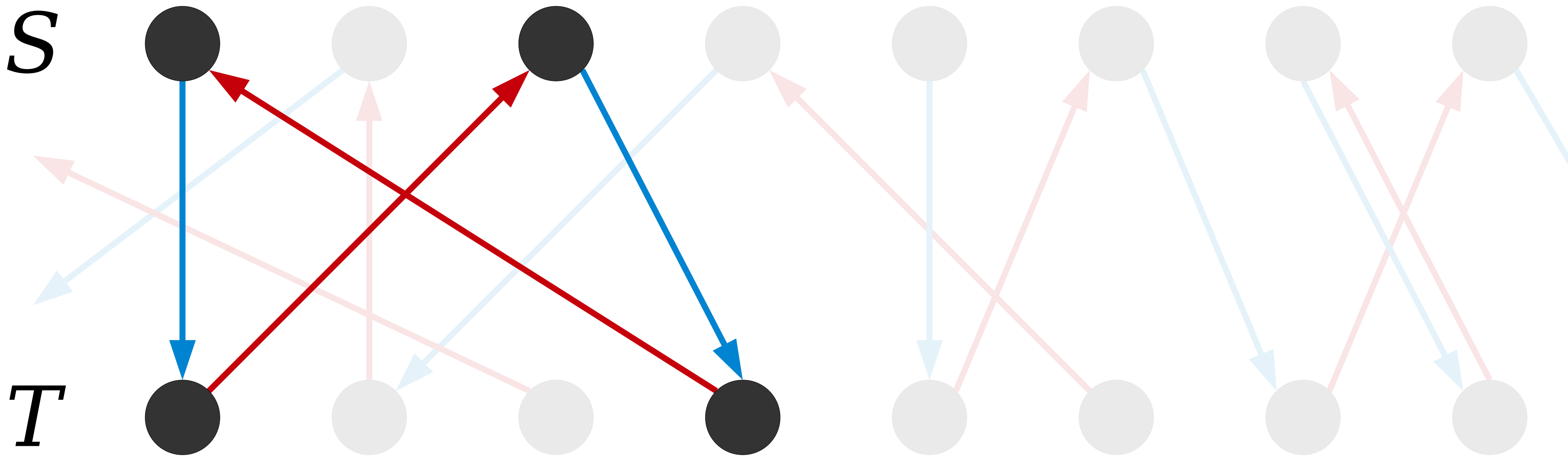
Theorem (Cantor-Bernstein-Schroeder): Let S and T be sets. If there is an injection $f : S \to T$ and an injection $g : T \to S$, then there is a bijection $h : S \to T$.



Blue lines represent the injection $f : S \to T$
Red lines represent the injection $g : T \to S$

Every node in this (possibly infinite) digraph has outdegree one and indegree at most one. Therefore, the digraph consists of a mix of paths and cycles.

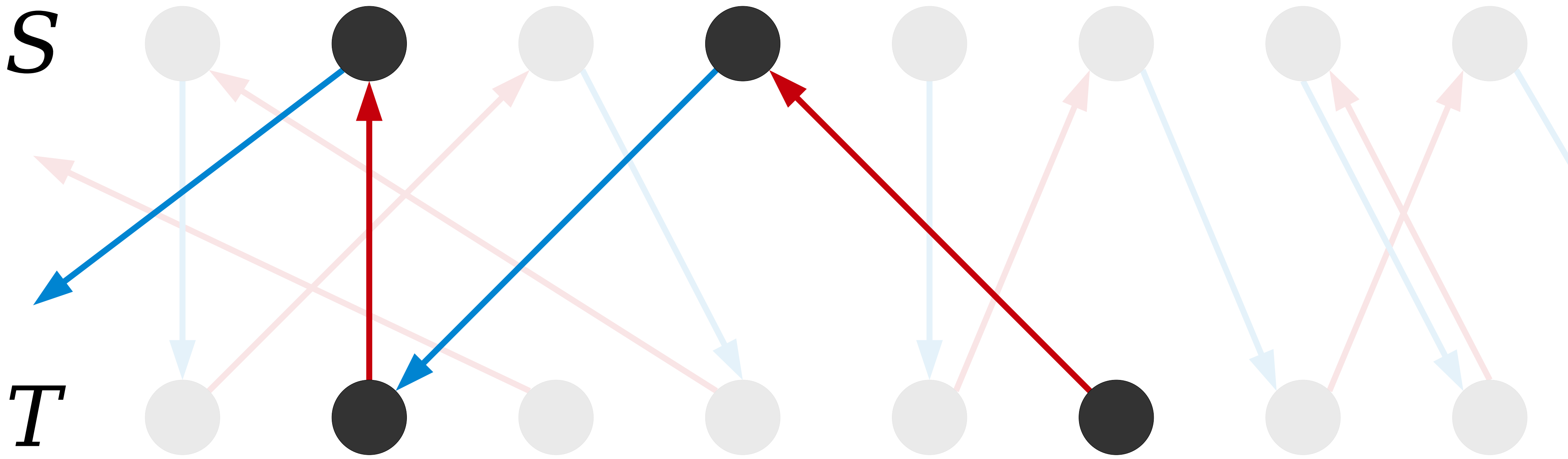
Theorem (Cantor-Bernstein-Schroeder): Let S and T be sets. If there is an injection $f : S \to T$ and an injection $g : T \to S$, then there is a bijection $h : S \to T$.



Blue lines represent the injection $f : S \to T$
Red lines represent the injection $g : T \to S$

For nodes within a cycle, define the bijection from S to T to be "follow the blue arrows."

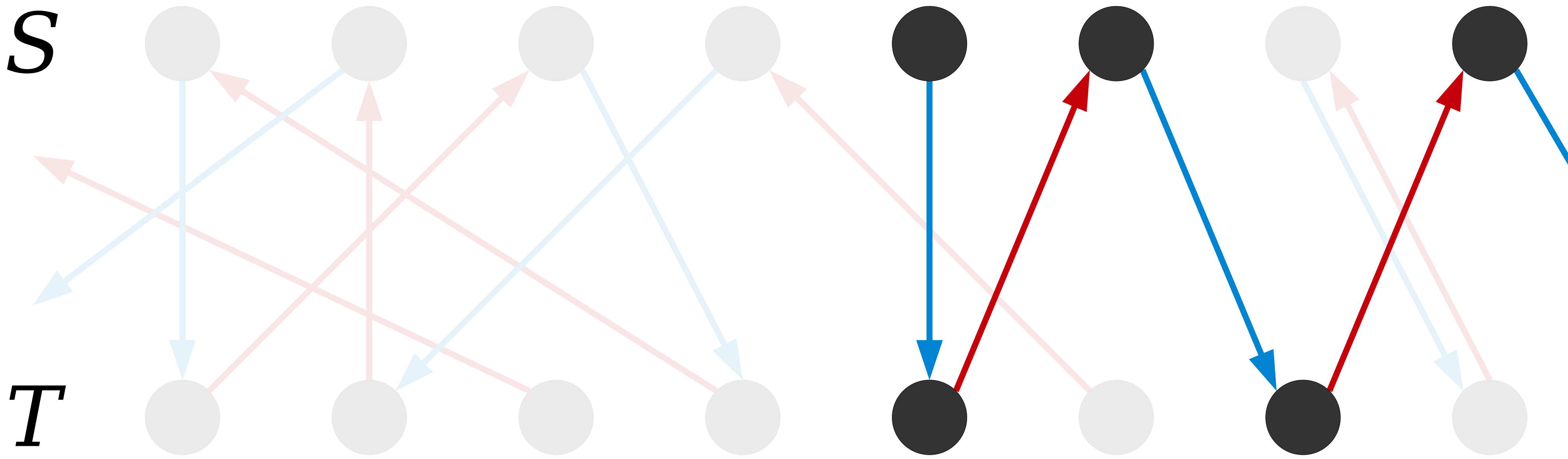
Theorem (Cantor-Bernstein-Schroeder): Let S and T be sets. If there is an injection $f : S \to T$ and an injection $g : T \to S$, then there is a bijection $h : S \to T$.



Blue lines represent the injection $f : S \to T$
Red lines represent the injection $g : T \to S$

For nodes in a path starting at a red node, have the bijection from S to T be "follow the red arrows in reverse."

Theorem (Cantor-Bernstein-Schroeder): Let S and T be sets. If there is an injection $f : S \to T$ and an injection $g : T \to S$, then there is a bijection $h : S \to T$.



Blue lines represent the injection $f : S \to T$
Red lines represent the injection $g : T \to S$

For nodes in any other path, have the bijection from S to T be "follow the blue arrows."

Theorem (Cantor-Bernstein-Schroeder): Let S and T be sets. If there is an injection $f : S \to T$ and an injection $g : T \to S$, then there is a bijection $h : S \to T$.

Recap for Today

- We can use ***walks*** and ***closed walks*** to travel around a graph. Walks and closed walks that don't repeat nodes or edges are called ***paths*** and ***cycles***, respectively.
- The ***complement*** of a graph is a graph formed by toggling which edges are included and which are excluded. At least one of a graph and its complement will always be connected.
- The ***indegree*** and ***outdegree*** of a node in a digraph are the number of edges entering or leaving the node, respectively.
- Digraphs where the indegree and outdegree of each node are at most one break apart into isolated paths and cycles.
- You can't trap a train on a track with teleporters, unless there's a teleporter behind the train.
- The Cantor-Bernstein-Schroeder theorem about sets and set cardinality can be thought of as a theorem about graphs with low indegrees and outdegrees.

Next Time

- ***The Pigeonhole Principle***
  - A simple, powerful, versatile theorem.
- ***Graph Theory Party Tricks***
  - Applying math to graphs of people!
- ***A Little Movie Puzzle***
  - Who watched what?