Pairing Algorithm: Calculating birth and death

Fudong Wang

Department of Mathematics and Statistics, University of South Florida

November 23, 2016

Basic terminology review

- Boundary ∂, Cycle c and Homology Group H_p
- Birth and Death

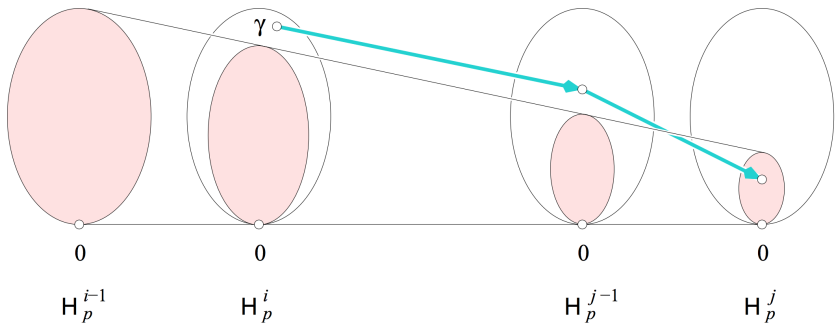


Figure 1: Birth and Death

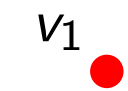
Positive and Negative

- Positive: simplex which create a non-boundary p-cycle
- Negative: simplex which kill an existing $(p-1)$-cycle

Flitration

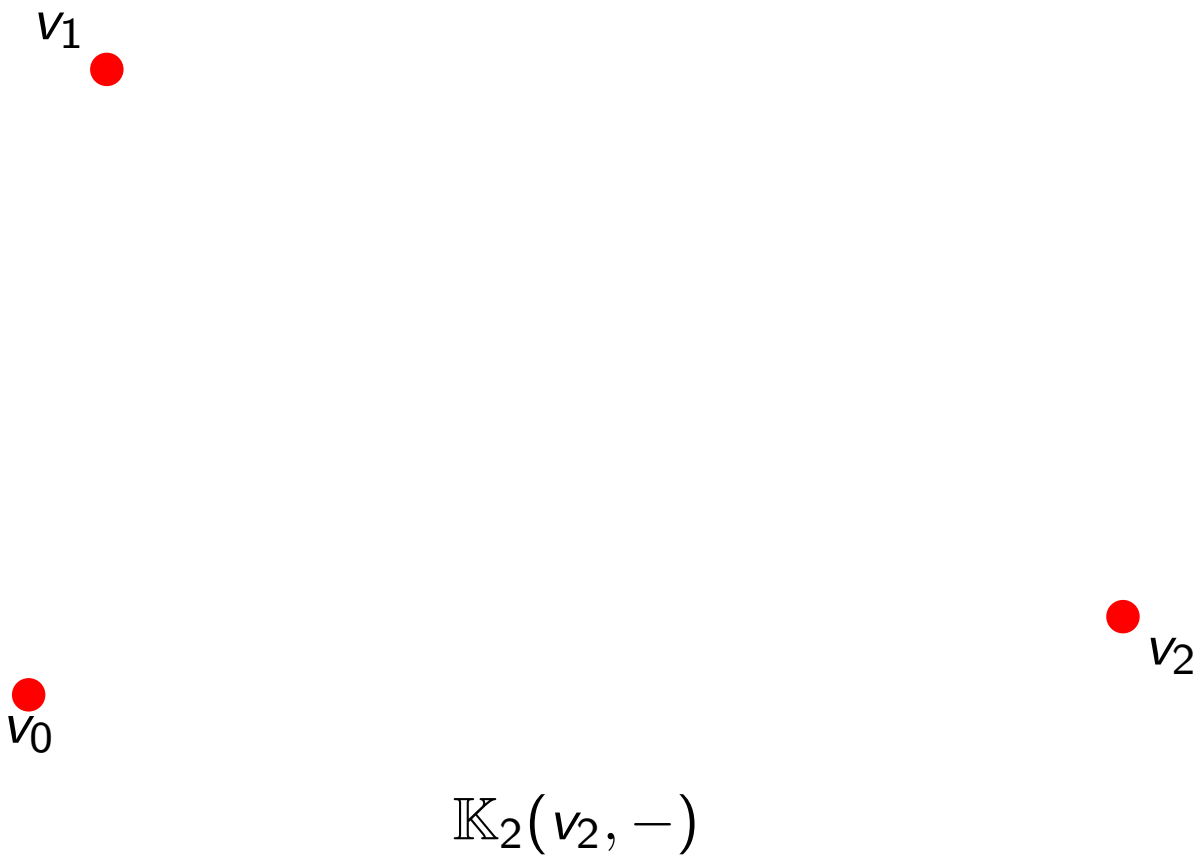
$$\mathbb{K}_0(v_0, -)$$

Flitration



$$\mathbb{K}_1(v_1, -)$$

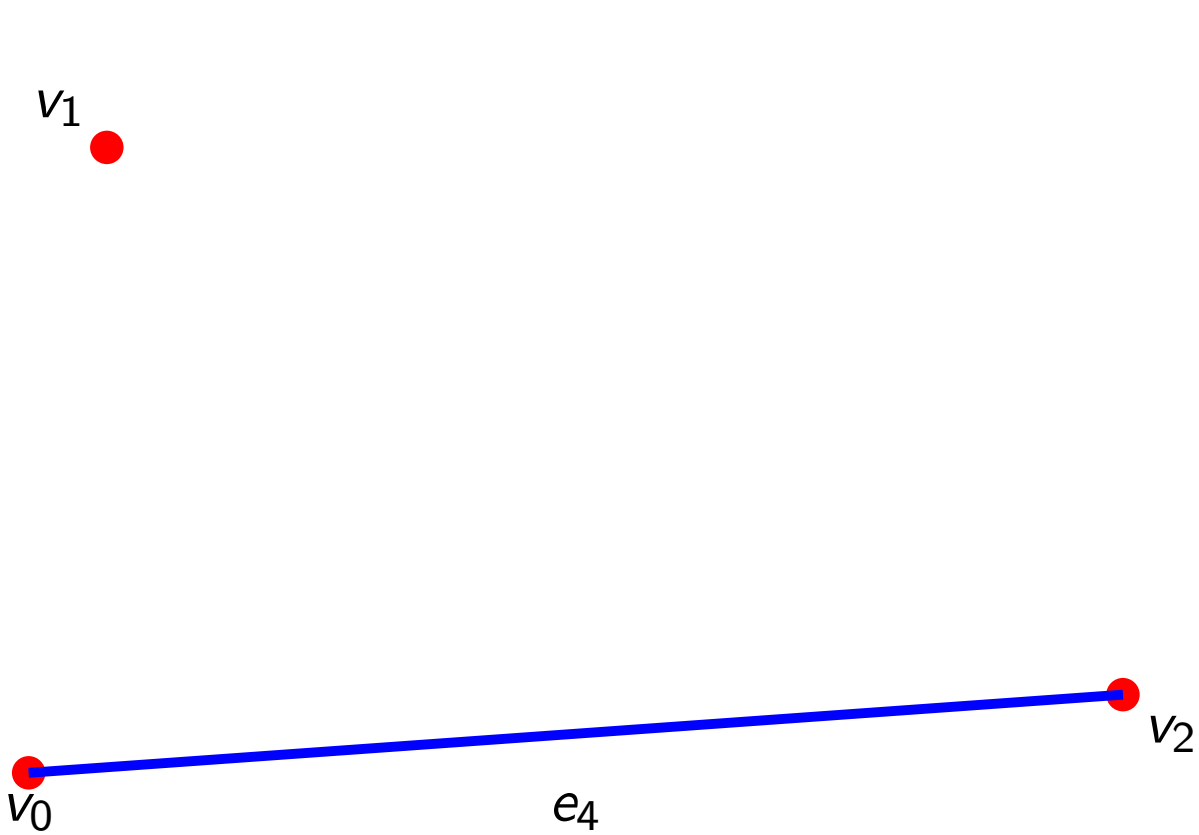
Flitration



Flitration

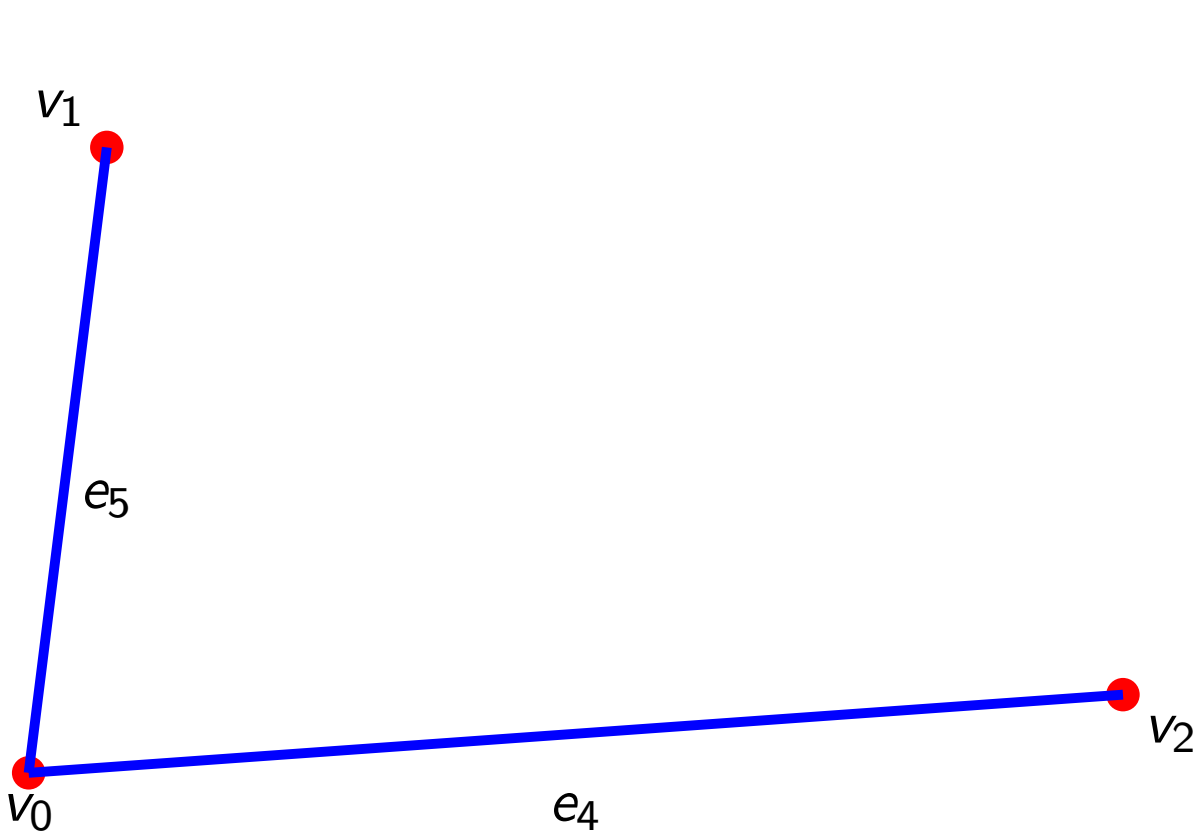
$\mathbb{K}_3(v_3, -)$

Flitration



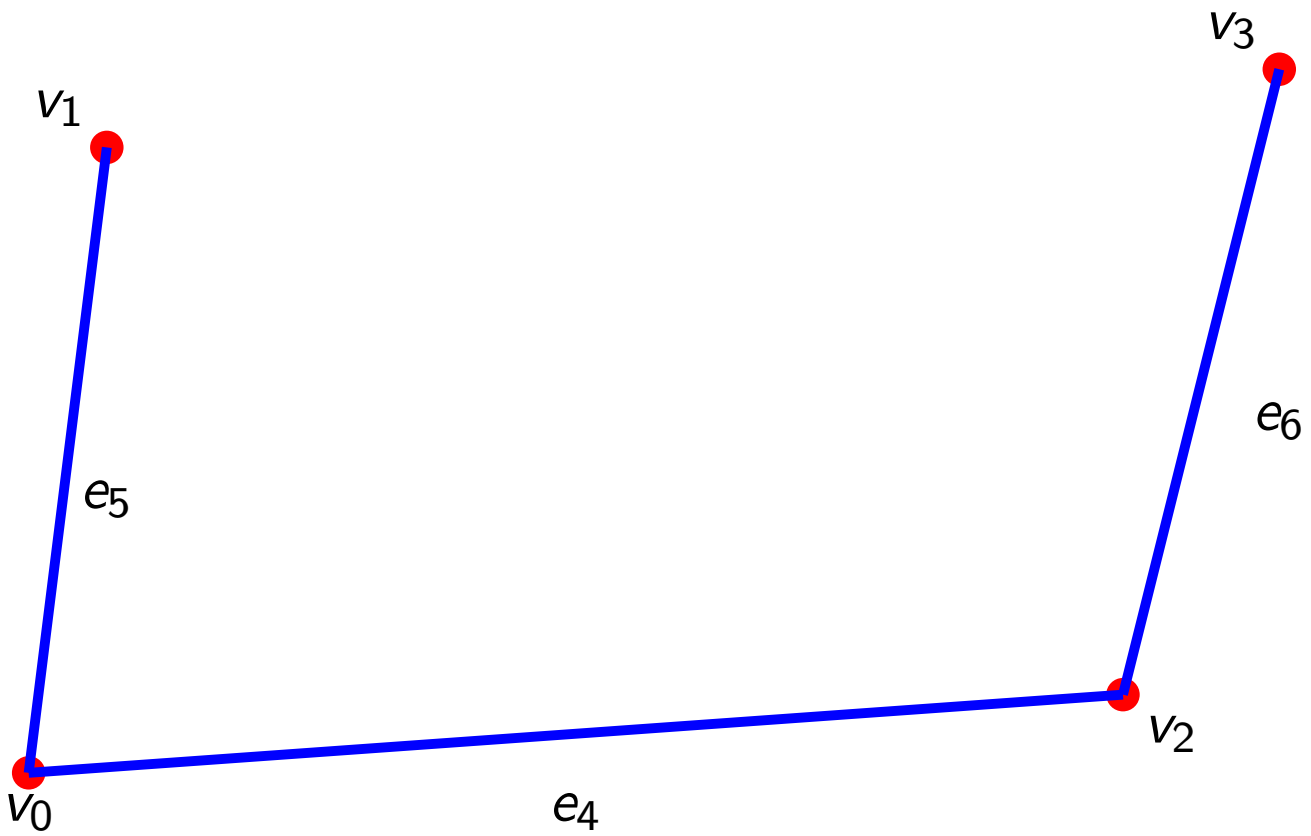
$$\mathbb{K}_4(v_2, e_4)$$

Flitration



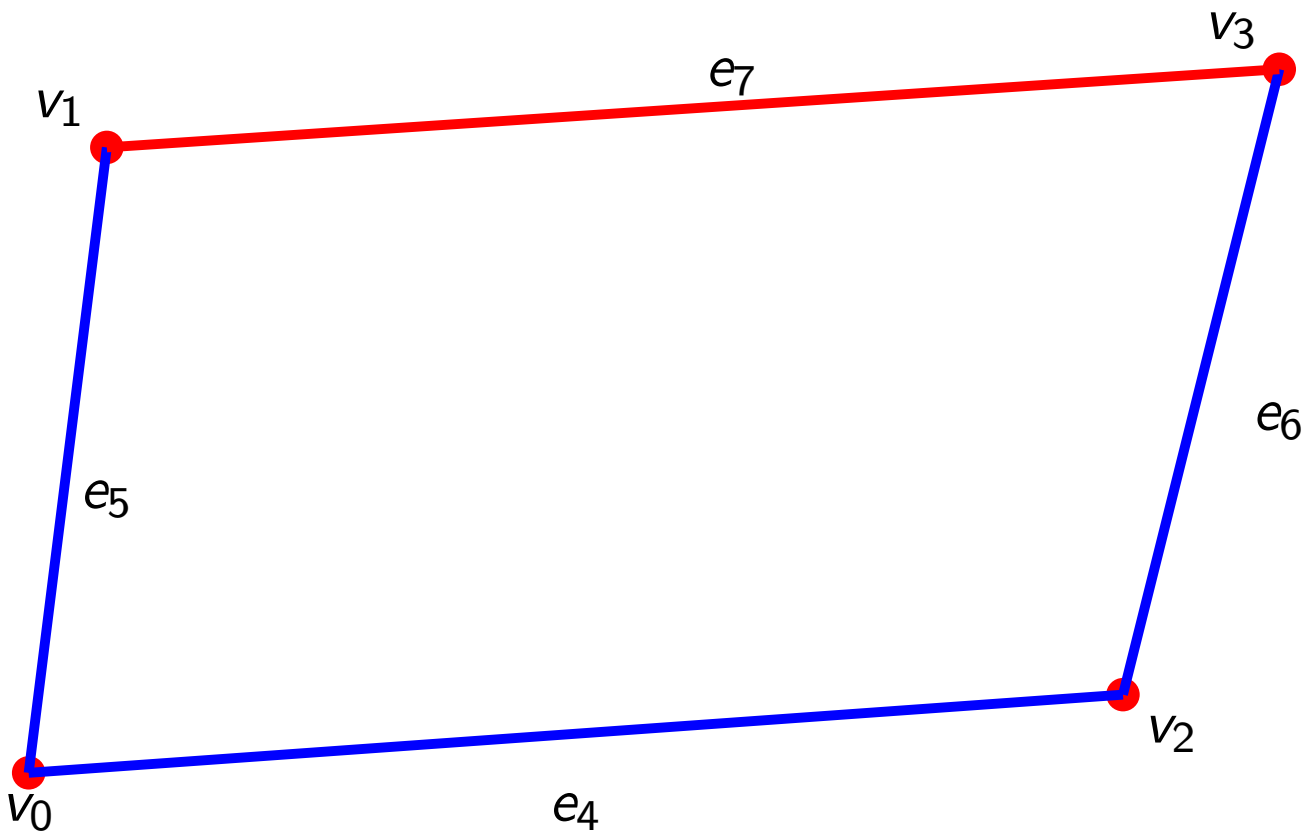
$\mathbb{K}_5(v_1, e_5)$

Flitration



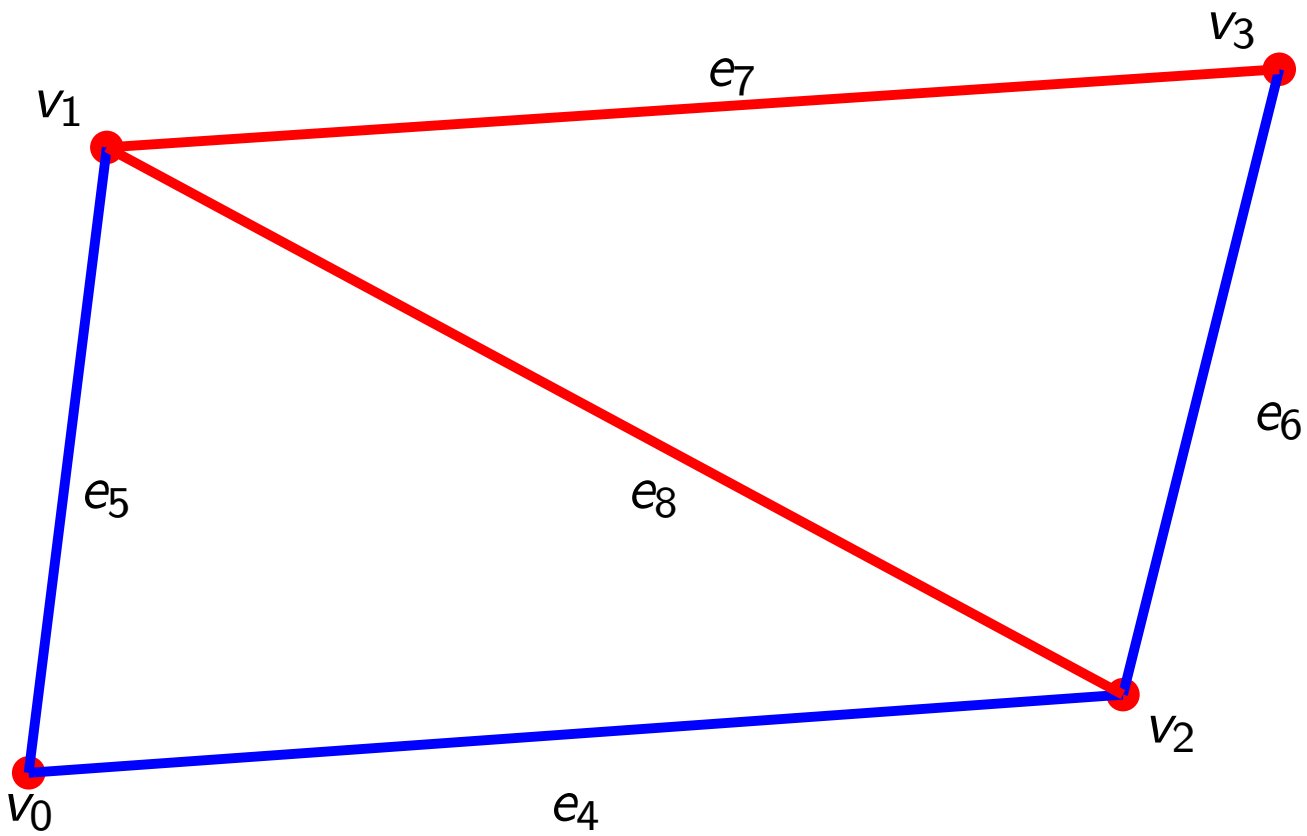
$\mathbb{K}_6(v_3, e_6)$

Flitration



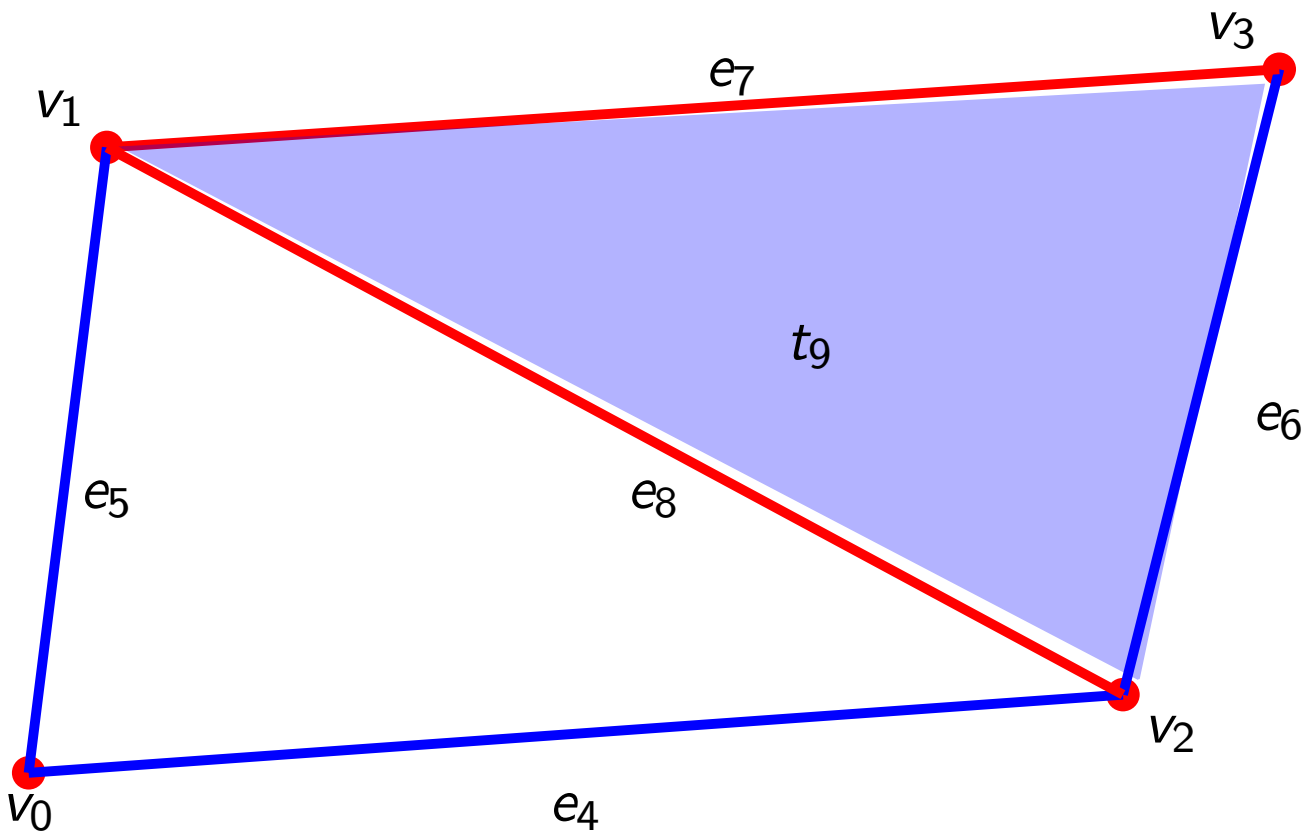
$$\mathbb{K}_7(e_7, -)$$

Flitration



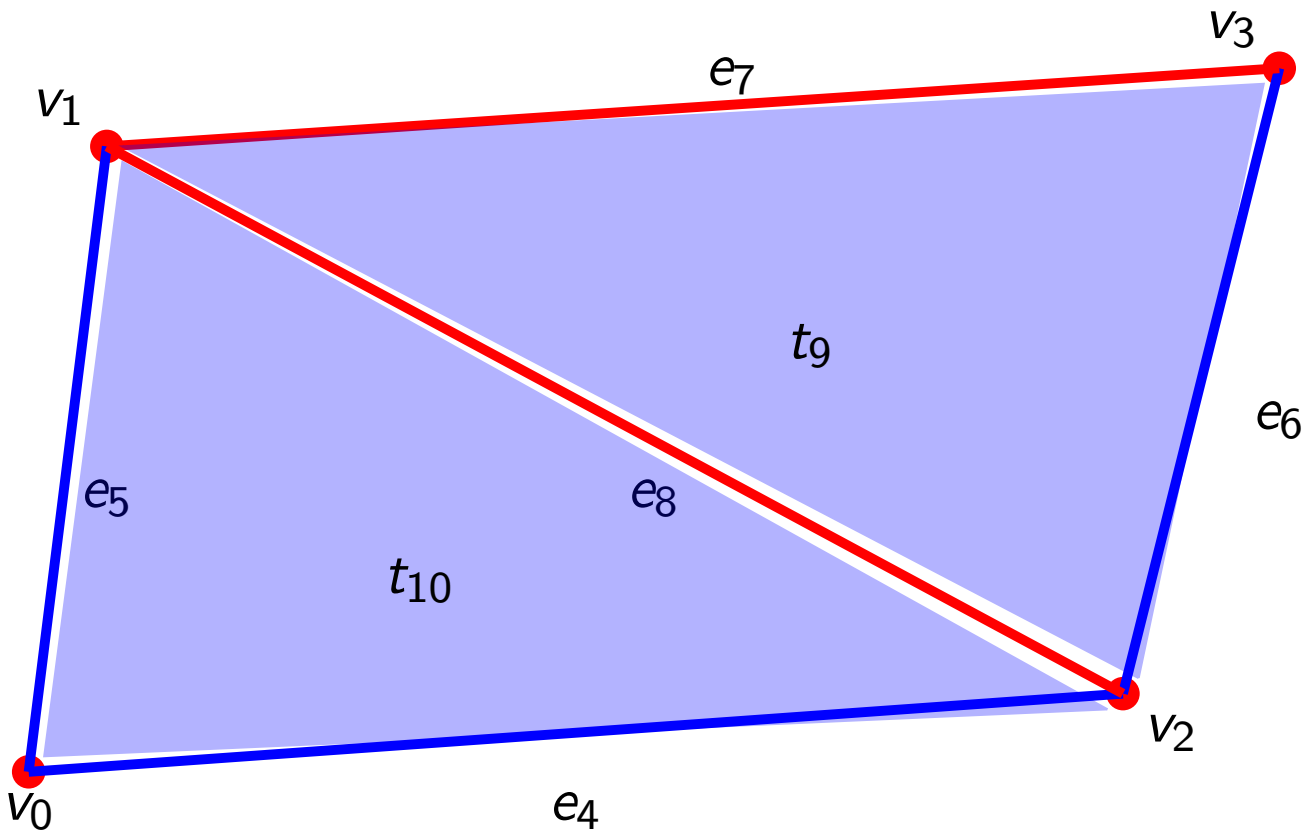
$$\mathbb{K}_8(e_8, -)$$

Flitration



$$\mathbb{K}_9(e_8, t_9)$$

Flitration



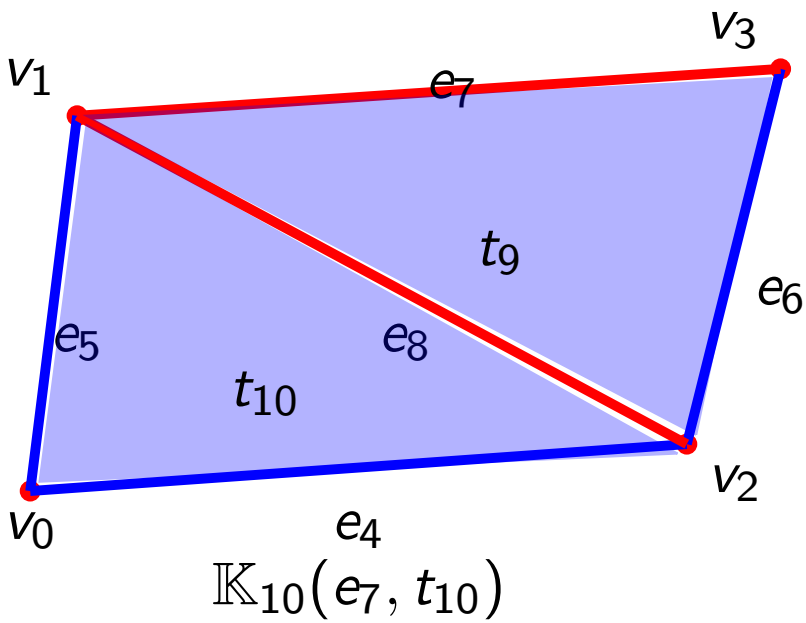
$$\mathbb{K}_{10}(e_7, t_{10})$$

Pairing Algorithm

Algorithm 1 PAIR(σ)

1: $c = \partial_p \sigma$
2: $d \leftarrow$ youngest positive $(p-1)$-simpliex $\in c$
3: **while** d is paired and c is not empty **do**
4: $\quad c'$ be the cycle killed by the simplex paired with d
5: $\quad c = c + c' \mod 2$
6: $\quad d \leftarrow$ youngest positive $(p-1)$-simpliex $\in c$
7: **end while**
8: **if** c is not empty **then**
9: $\quad \sigma$ is a negative p-simplex paired with d
10: **else**
11: $\quad \sigma$ is a positive p-simplex
12: **end if**

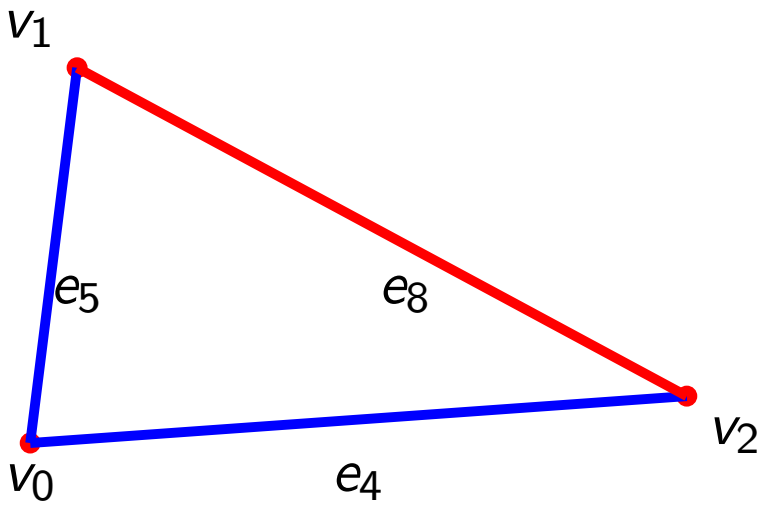
Pairing Algorithm: Example



$\mathbb{K}_{10}(e_7, t_{10})$

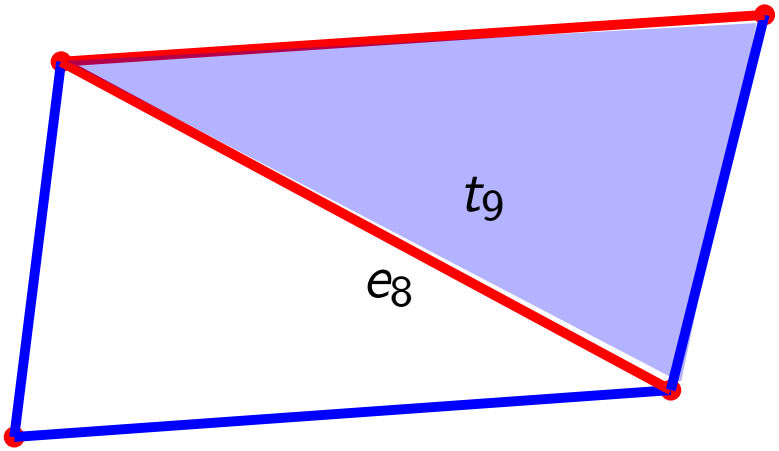
- $c = \partial_p(t_{10}) = e_5 + e_4 + e_8$

Pairing Algorithm: Example



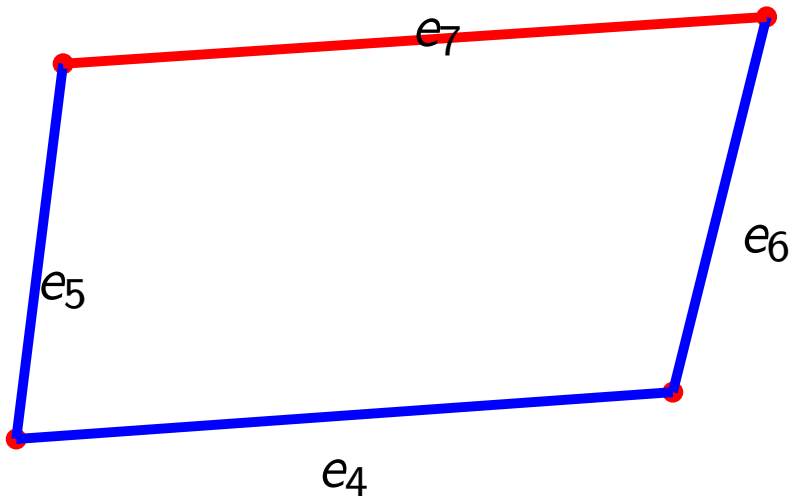
- $c = \partial_p(t_{10}) = e_5 + e_4 + e_8$
- $d = e_8$, the red edge

Pairing Algorithm: Example



- $c = \partial_p(t_{10}) = e_5 + e_4 + e_8$
- $d = e_8$, the red edge
- Into while-do loop since e_8 pair with t_9

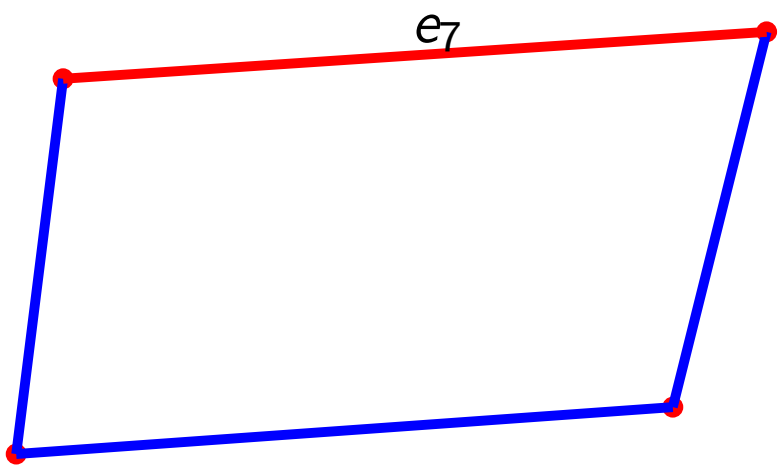
Pairing Algorithm: Example



- $c = \partial_p(t_{10}) = e_5 + e_4 + e_8$
- $d = e_8$, the red edge
- Into while-do loop since e_8 pair with t_9
- Update c

$$\begin{aligned} c &= e_5 + e_4 + e_8 + \partial_p(t_9) \\ &= e_4 + e_5 + e_6 + e_7 \end{aligned}$$

Pairing Algorithm: Example



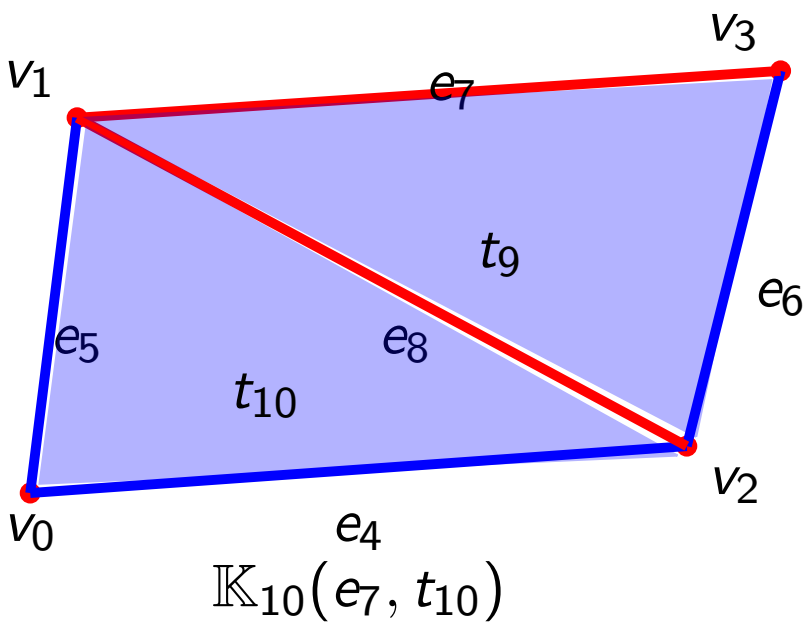
$\mathbb{K}_7(e_7, -)$

- $c = \partial_p(t_{10}) = e_5 + e_4 + e_8$
- $d = e_8$, the red edge
- Into while-do loop since e_8 pair with t_9
- Update c

$$\begin{aligned} c &= e_5 + e_4 + e_8 + \partial_p(t_9) \\ &= e_4 + e_5 + e_6 + e_7 \end{aligned}$$

- $d = e_7$,Not Paired

Pairing Algorithm: Example



$\mathbb{K}_{10}(e_7, t_{10})$

- $c = \partial_p(t_{10}) = e_5 + e_4 + e_8$
- $d = e_8$, the red edge
- Into while-do loop since e_8 pair with t_9
- Update c

$$c = e_5 + e_4 + e_8 + \partial_p(t_9)$$
$$= e_4 + e_5 + e_6 + e_7$$

- $d = e_7$,Not Paired
- t_{10} is negative paired with e_7

Implementation by Python

Data structure (dict): Input

```
1  K={x:[[],[]] for x in range(11)}
2  sigma = {0:[0],1:[1],2:[2],3:[3],...,10:[0,1,2]}
```

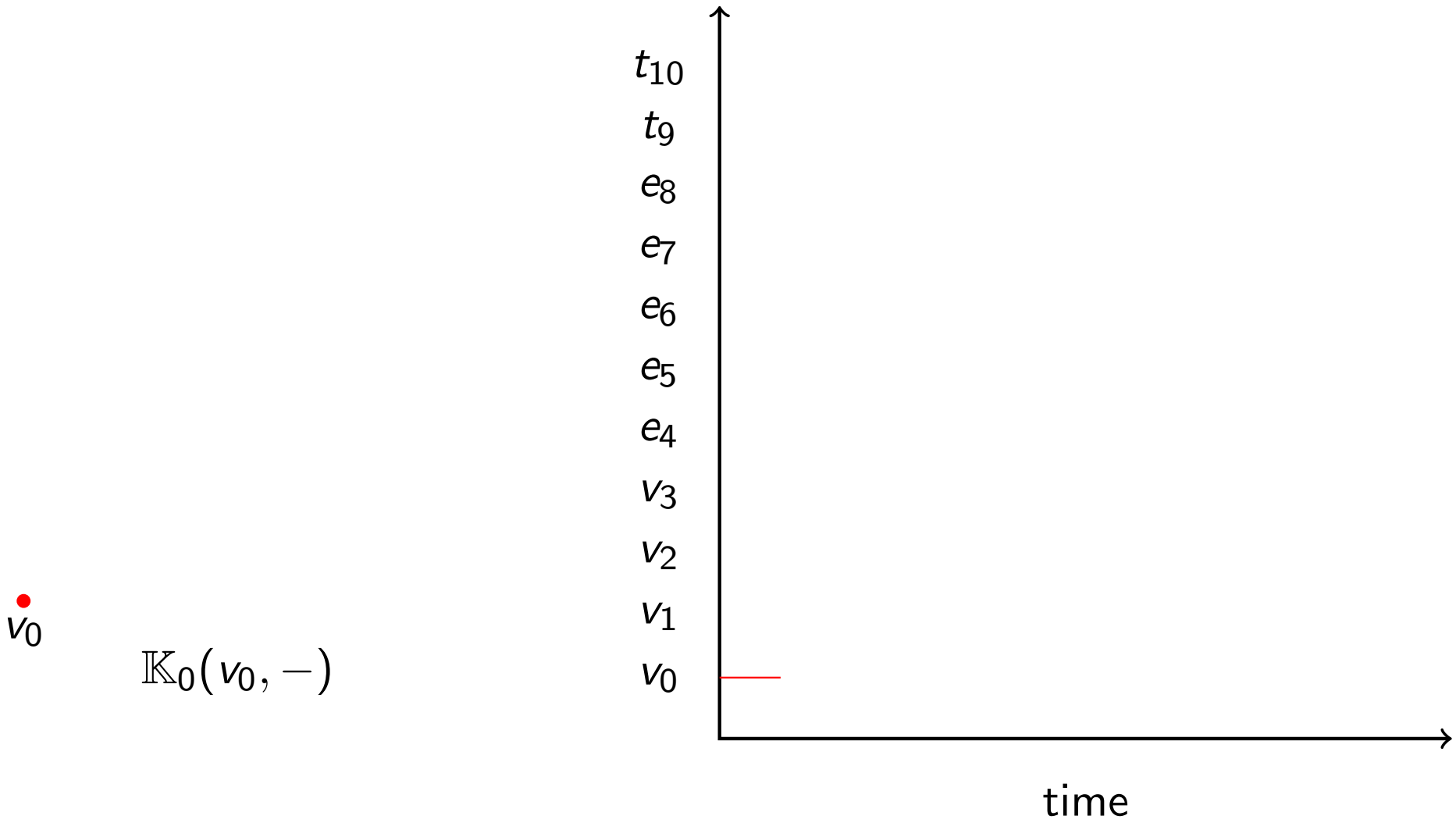
K is a dictionary, the key is the index of filtration, the corresponding value is a list : [positive, negative],
sigma is a dictionary, the key is the index of filtration, the corresponding value is a list: for 0-simplex:[a], 1-simplex: [a,b], 2-simplex: [a,b,c],etc.

Implementation by Python

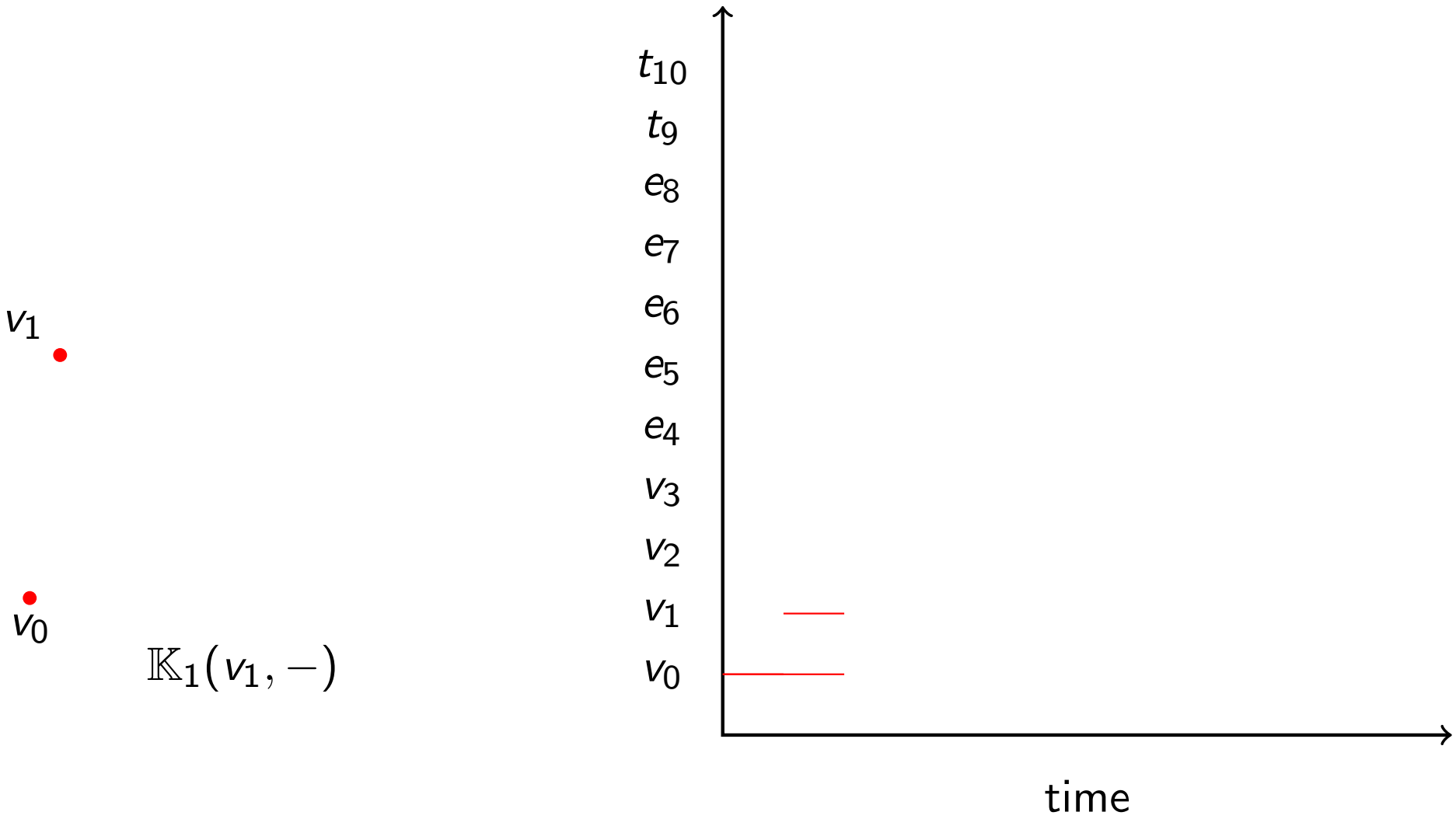
Data structure (dict): Output

```
{0: [[0], 'unpaired'],
 1: [[1], 'unpaired'],
 2: [[2], 'unpaired'],
 3: [[3], 'unpaired'],
 4: [[2], [4]],
 5: [[1], [5]],
 6: [[3], [6]],
 7: [[7], 'unpaired'],
 8: [[8], 'unpaired'],
 9: [[8], [9]],
 10: [[7], [10]]}
```

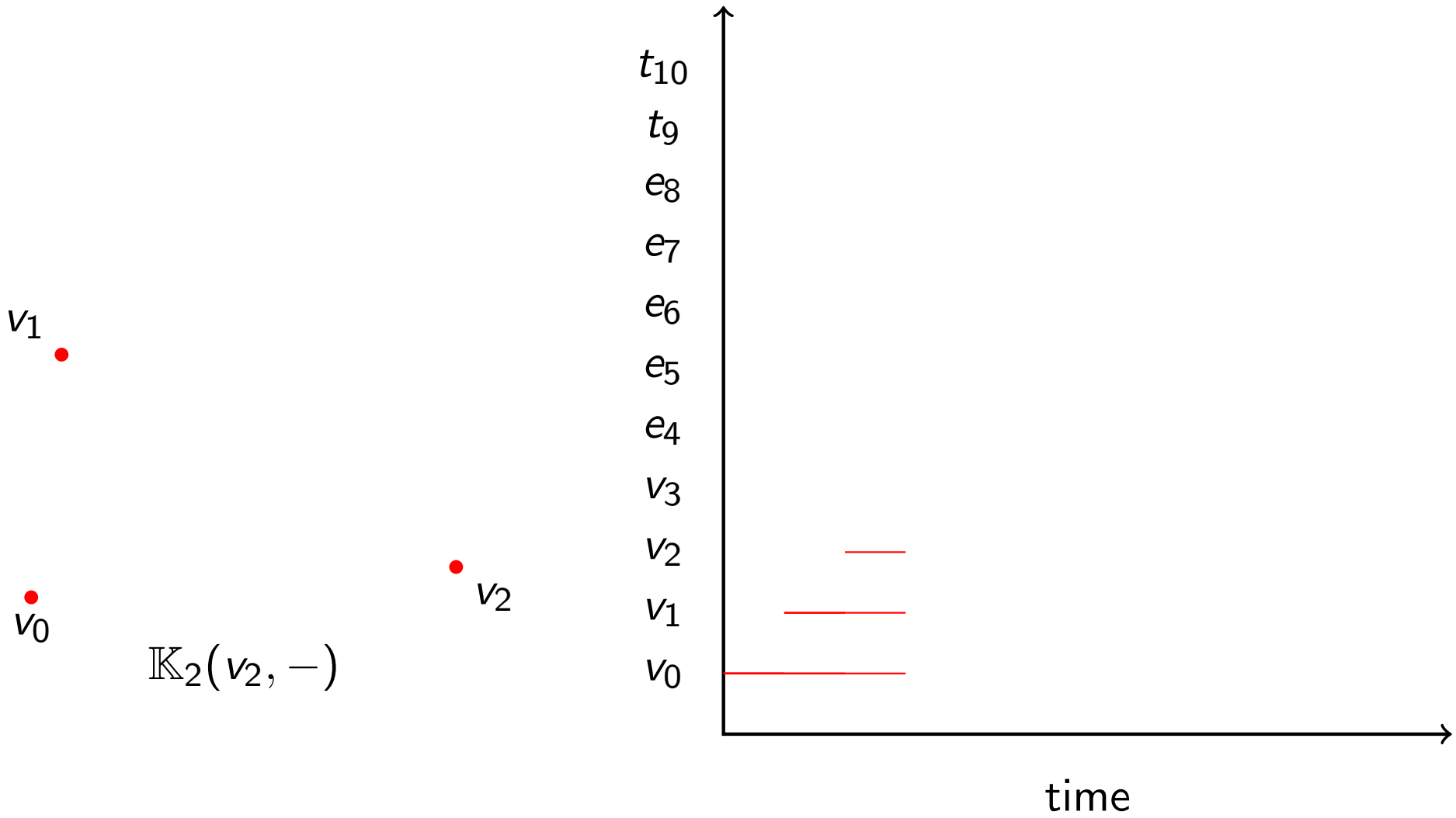
Birth and Death



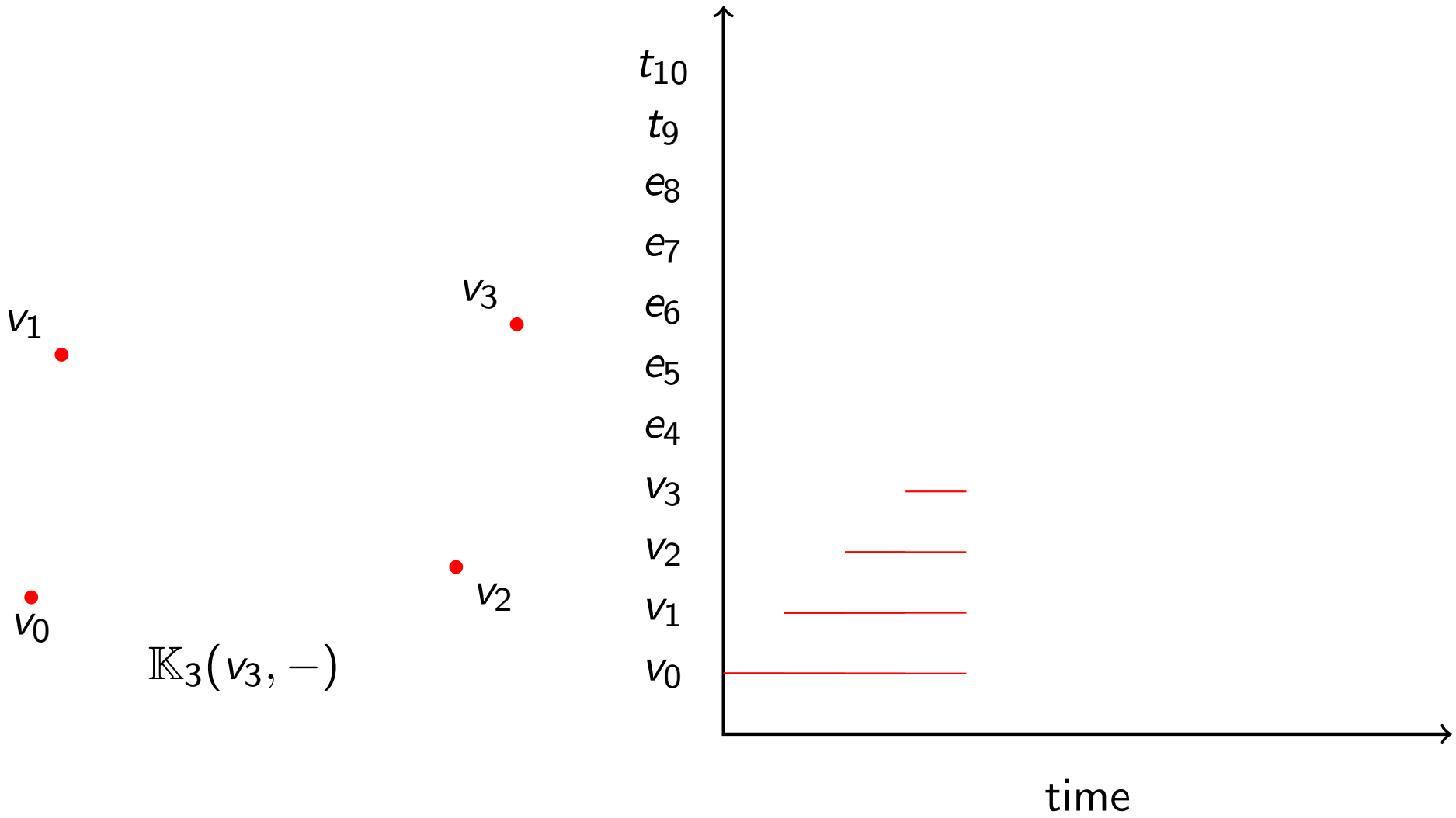
Birth and Death



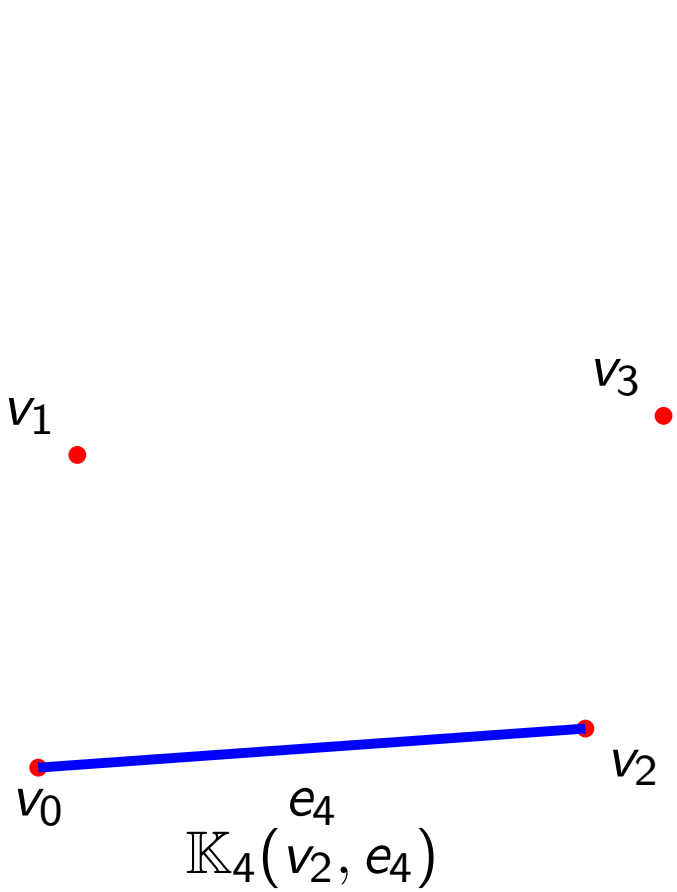
Birth and Death

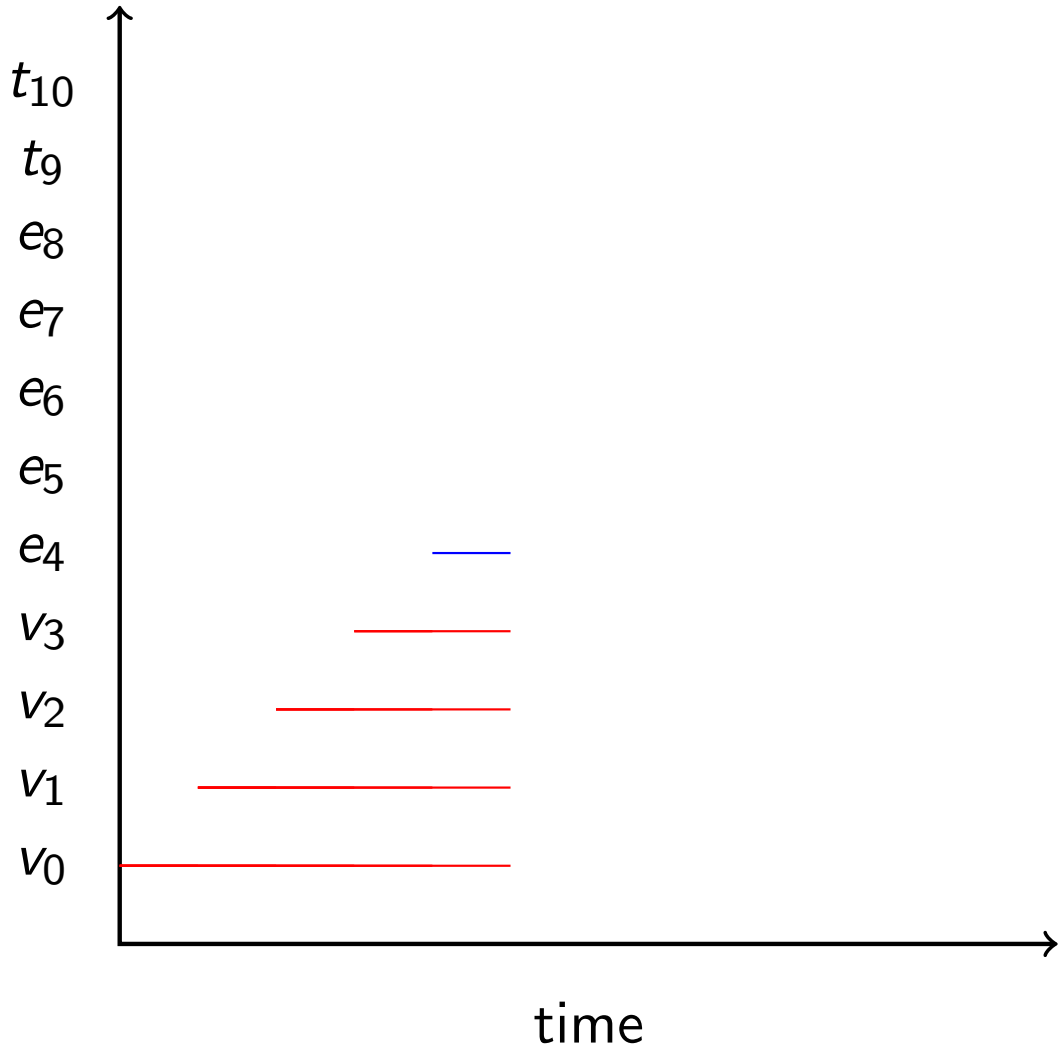


Birth and Death

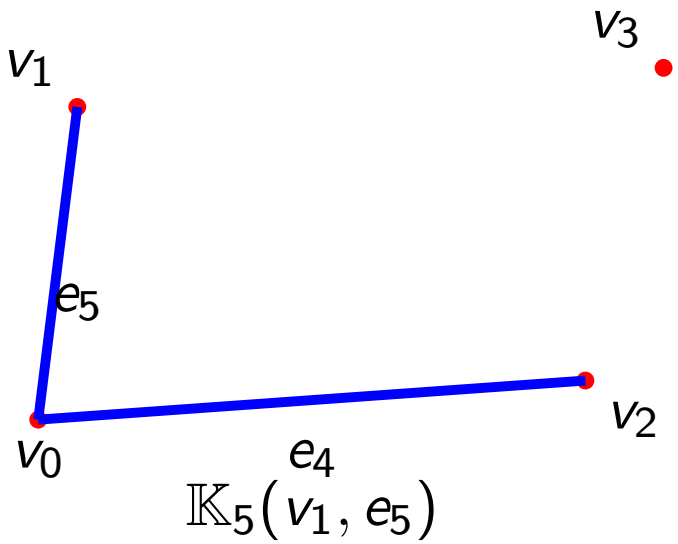


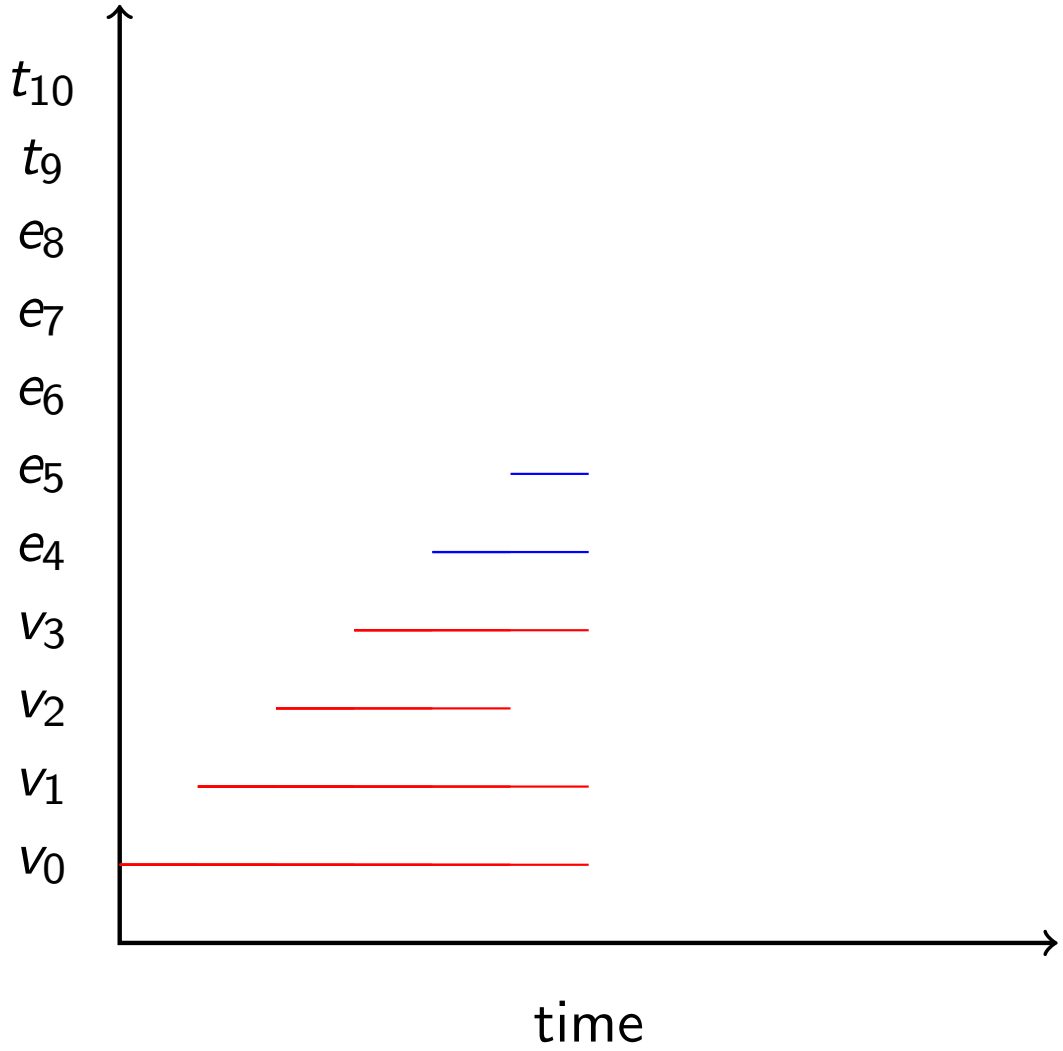
Birth and Death



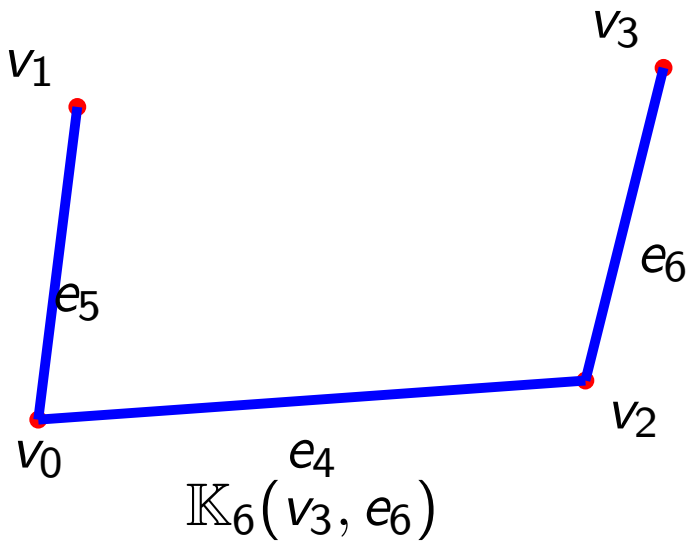


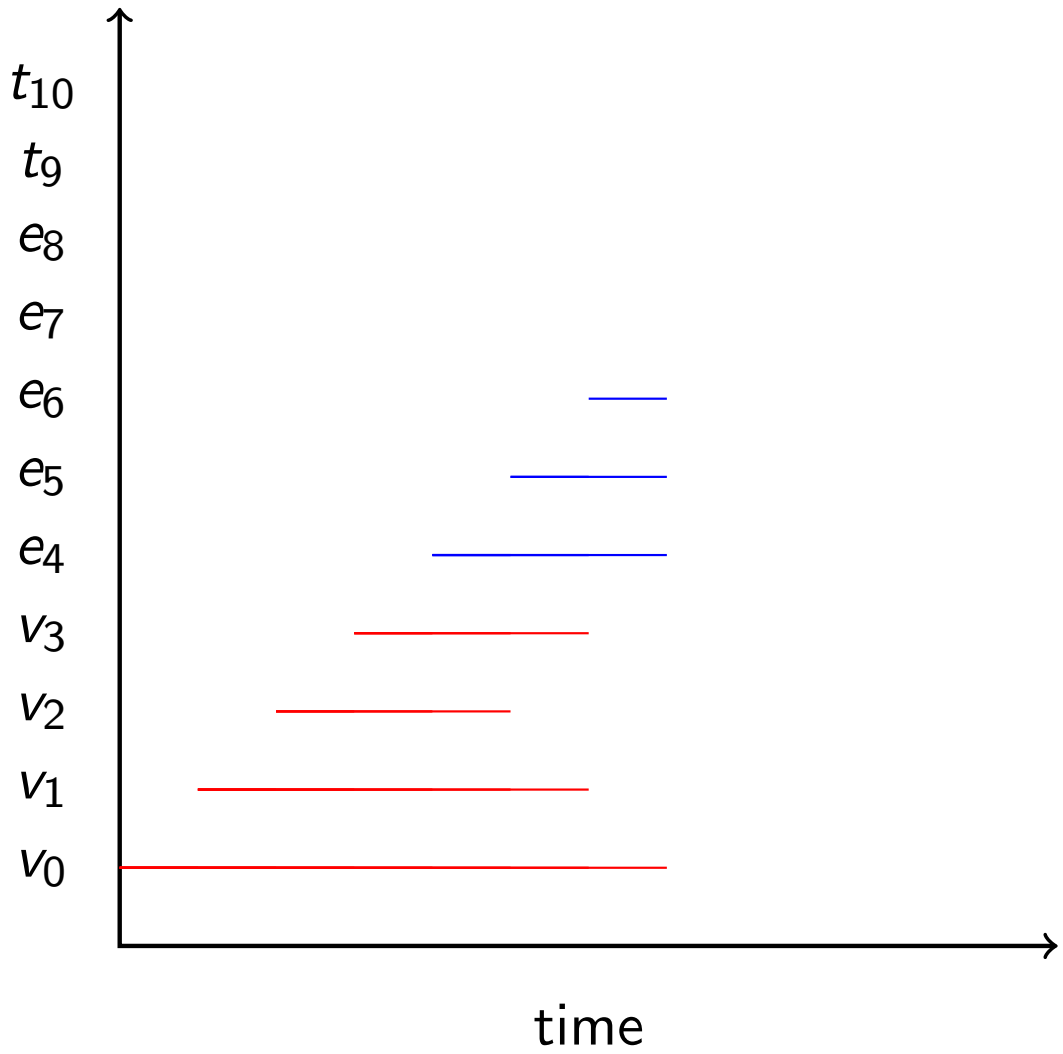
Birth and Death



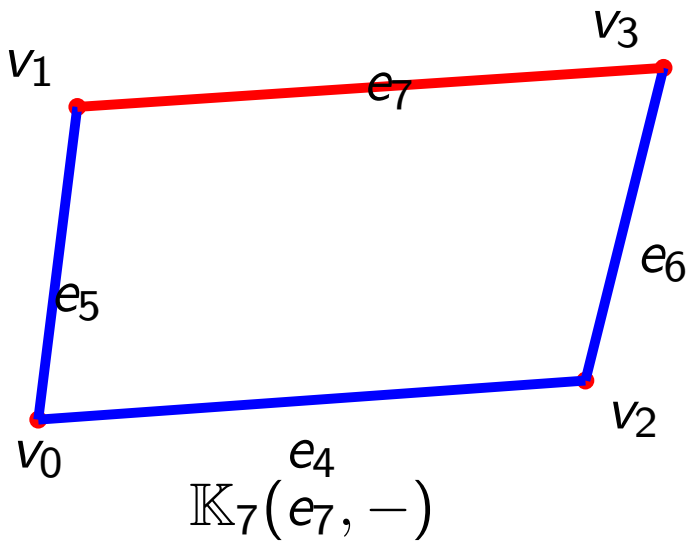


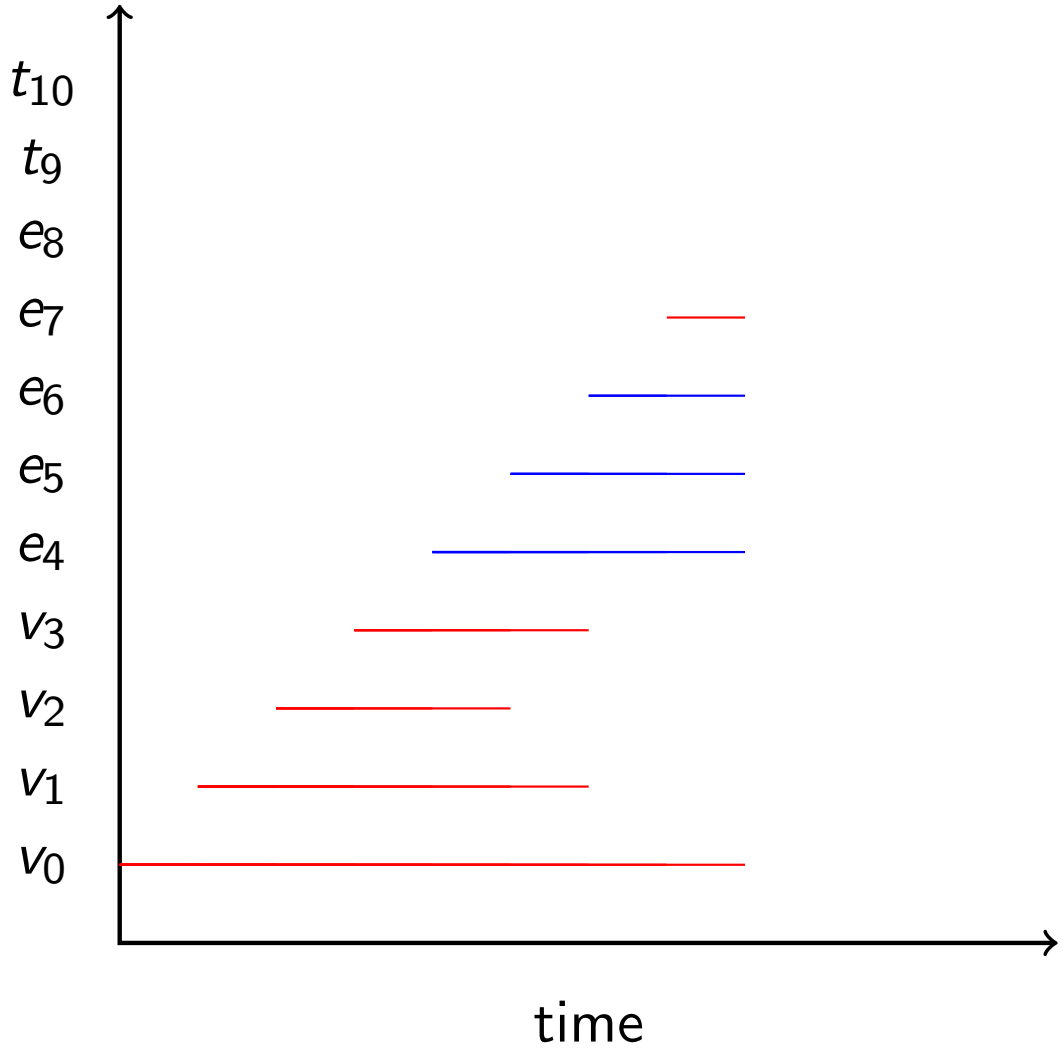
Birth and Death



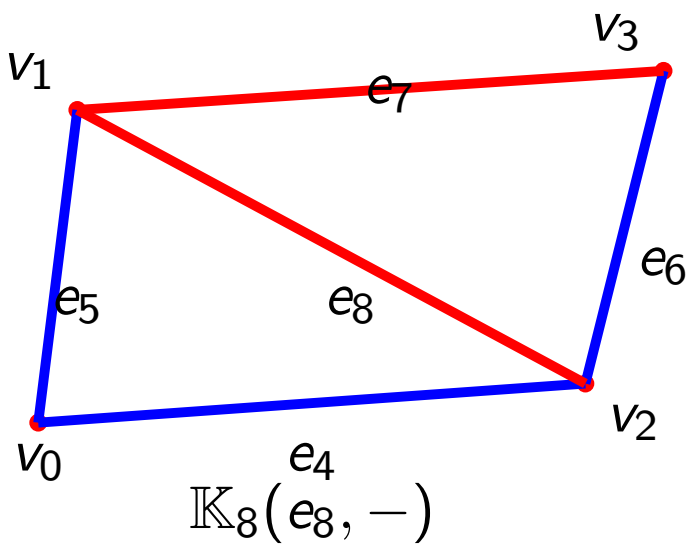


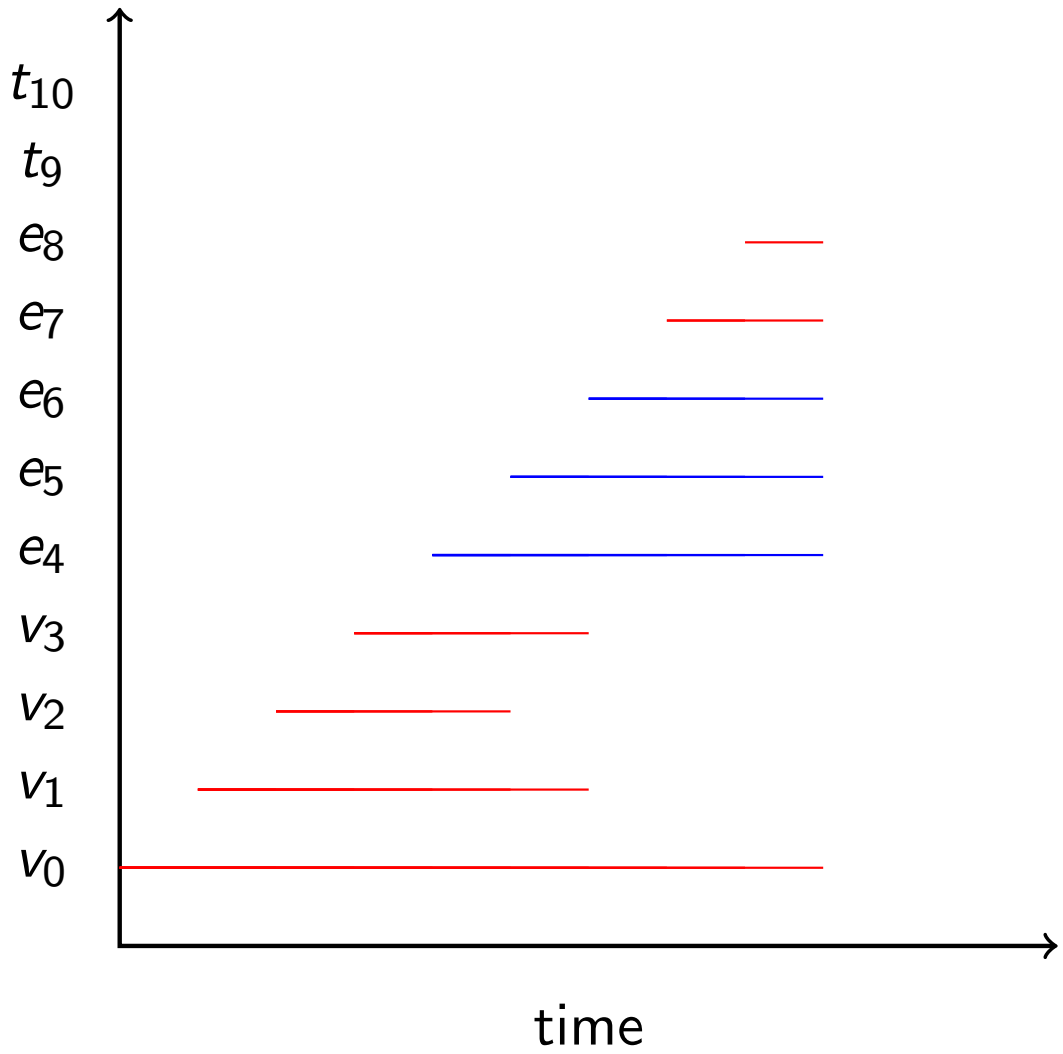
Birth and Death



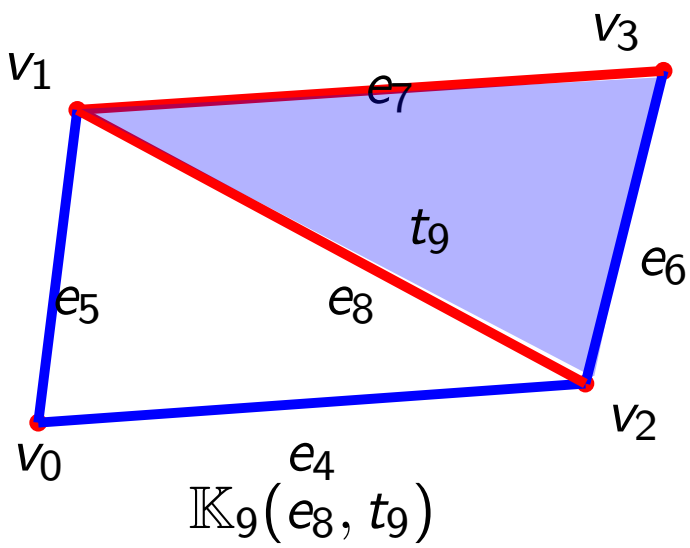


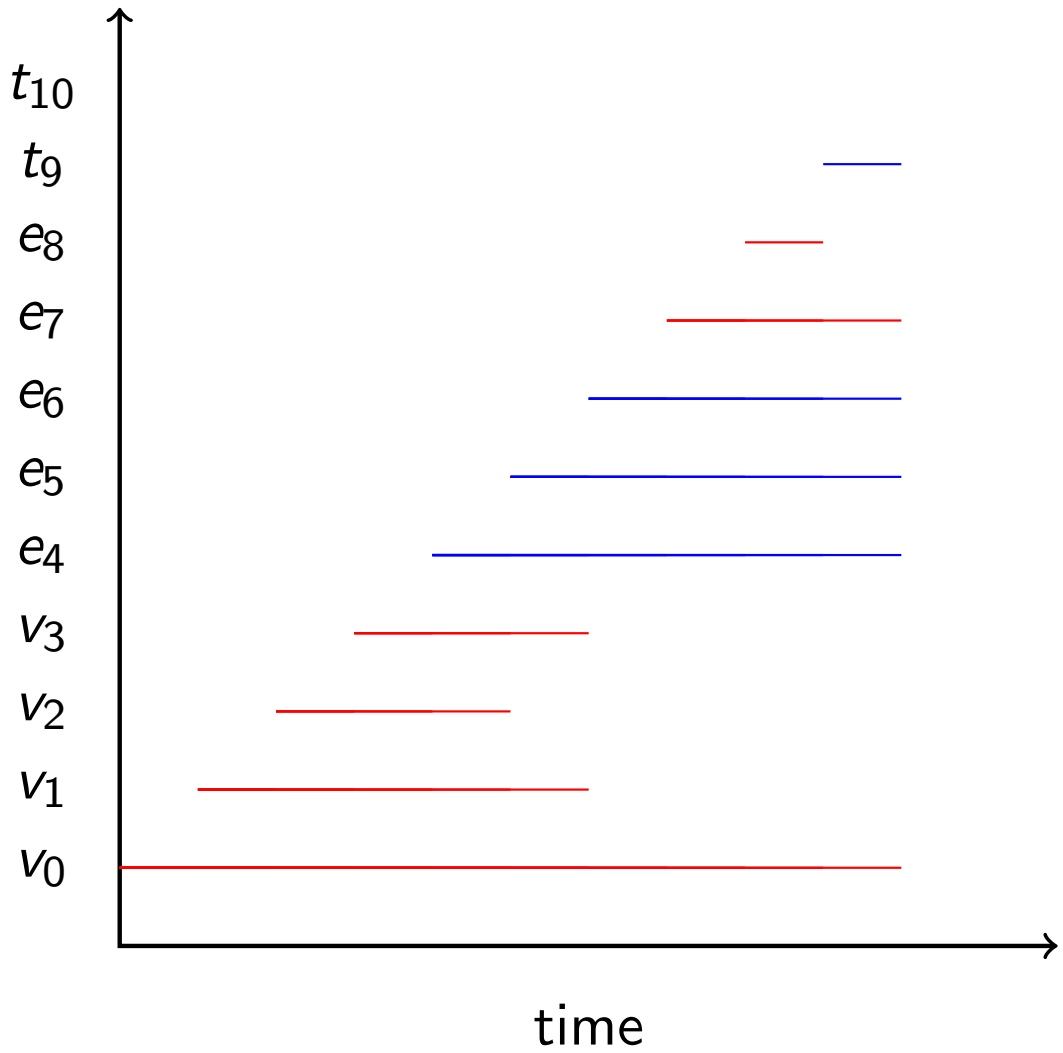
Birth and Death



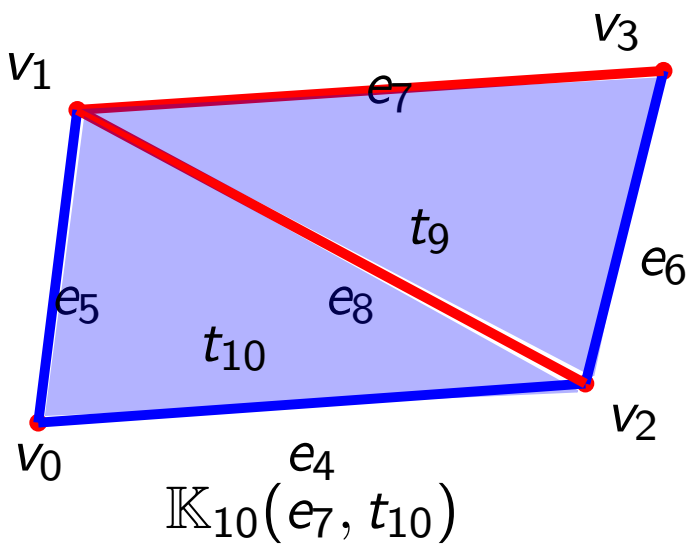


Birth and Death

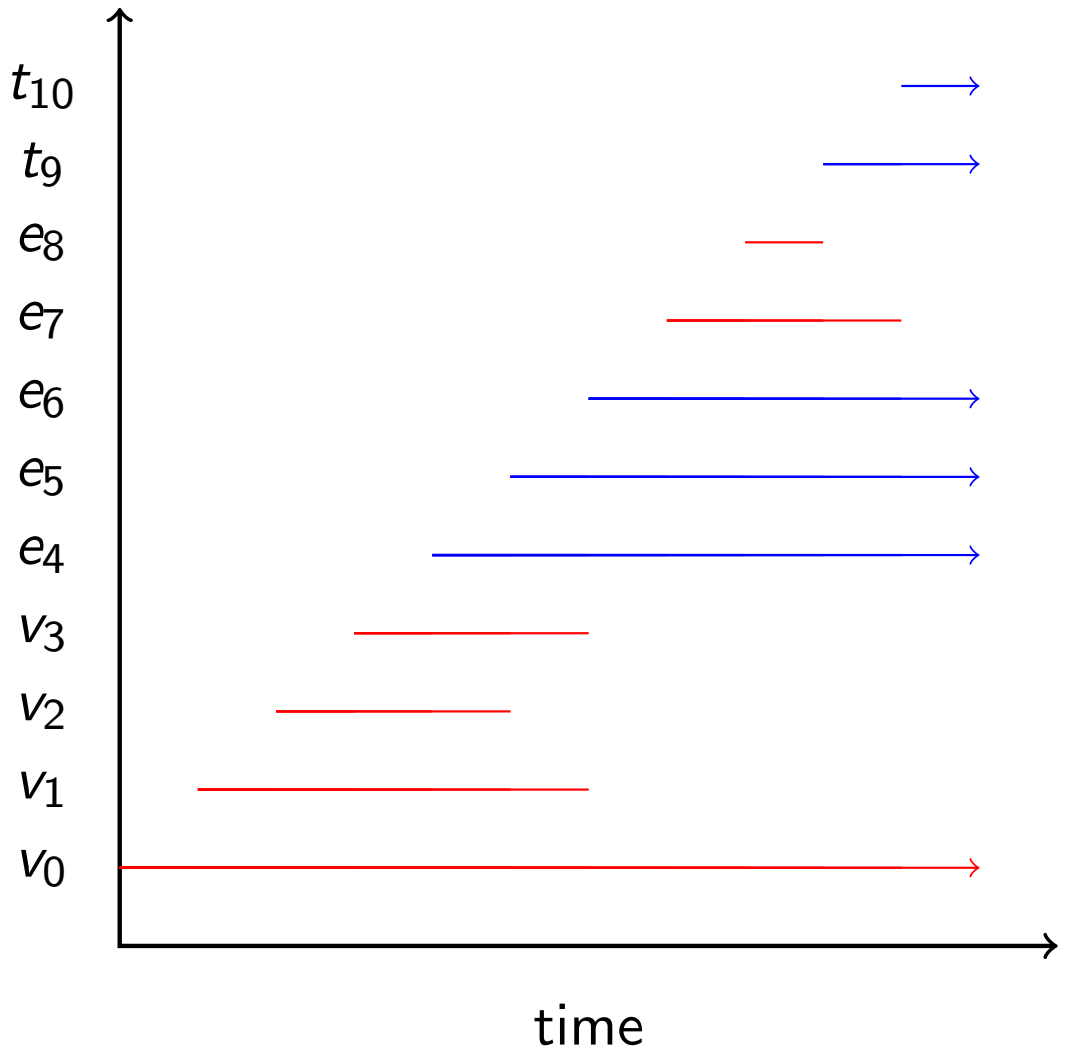




Birth and Death



$\mathbb{K}_{10}(e_7, t_{10})$



Homology Group