Logic Programming - Traveling

Assume we have the following facts:

```
feet(fhnw, olten_sbb).
feet(fhnw, olten_city).

train(olten_sbb, bern_sbb).
train(olten_sbb, zurich_airport).
train(olten_sbb, zurich_sbb).

flight(zurich_airport, las_vegas).

bus(las_vergas, yosemite).
bus(yosemite, san_francisco).
```

Exercise

Write Horn clauses defining the following predicates **`path/2`** with the obvious meaning, i.e. there is a path from one point to another, e.g.

```
?- path(fhnw, san_francisco).
```

Solution

```
feet(fhnw, olten_sbb).

train(olten_sbb, bern_sbb).
train(olten_sbb, zurich_airport).
train(olten_sbb, zurich_sbb).

flight(zurich_airport, las_vegas).

bus(las_vegas, yosemite).
bus(yosemite, san_francisco).

aPath(A,Z) :- feet(A,Z).
aPath(A,Z) :- train(A,Z).
aPath(A,Z) :- flight(A,Z).
aPath(A,Z) :- bus(A,Z).

path(A,Z) :- aPath(A,Z).
path(A,Z) :- aPath(A,B), path(B,Z).
```

However, the following solution DOES NOT WORK: it runs into an endless loop.

```
path(X,Y) :- feet(X,Y).
path(X,Y) :- train(X,Y).
path(X,Y) :- flight(X,Y).
path(X,Y) :- bus(X,Y).

path(X,Y) :- path(X,Z), path(Z,Y).
```

Based on the solution for path we can develop the solution for change …

```
feet(fhnw, olten_sbb).

train(olten_sbb, bern_sbb).
train(olten_sbb, zurich_airport).
train(olten_sbb, zurich_sbb).

flight(zurich_airport, las_vegas).

bus(las_vegas, yosemite).
bus(yosemite, san_francisco).

aChange(A,Z) :- feet(A,Z).
aChange(A,Z) :- train(A,Z).
aChange(A,Z) :- flight(A,Z).
aChange(A,Z) :- bus(A,Z).

change(A,Z,0) :- aChange(A,Z).
change(A,Z,C) :- aChange(A,B), change(B,Z,C1), C is C1 + 1.
```