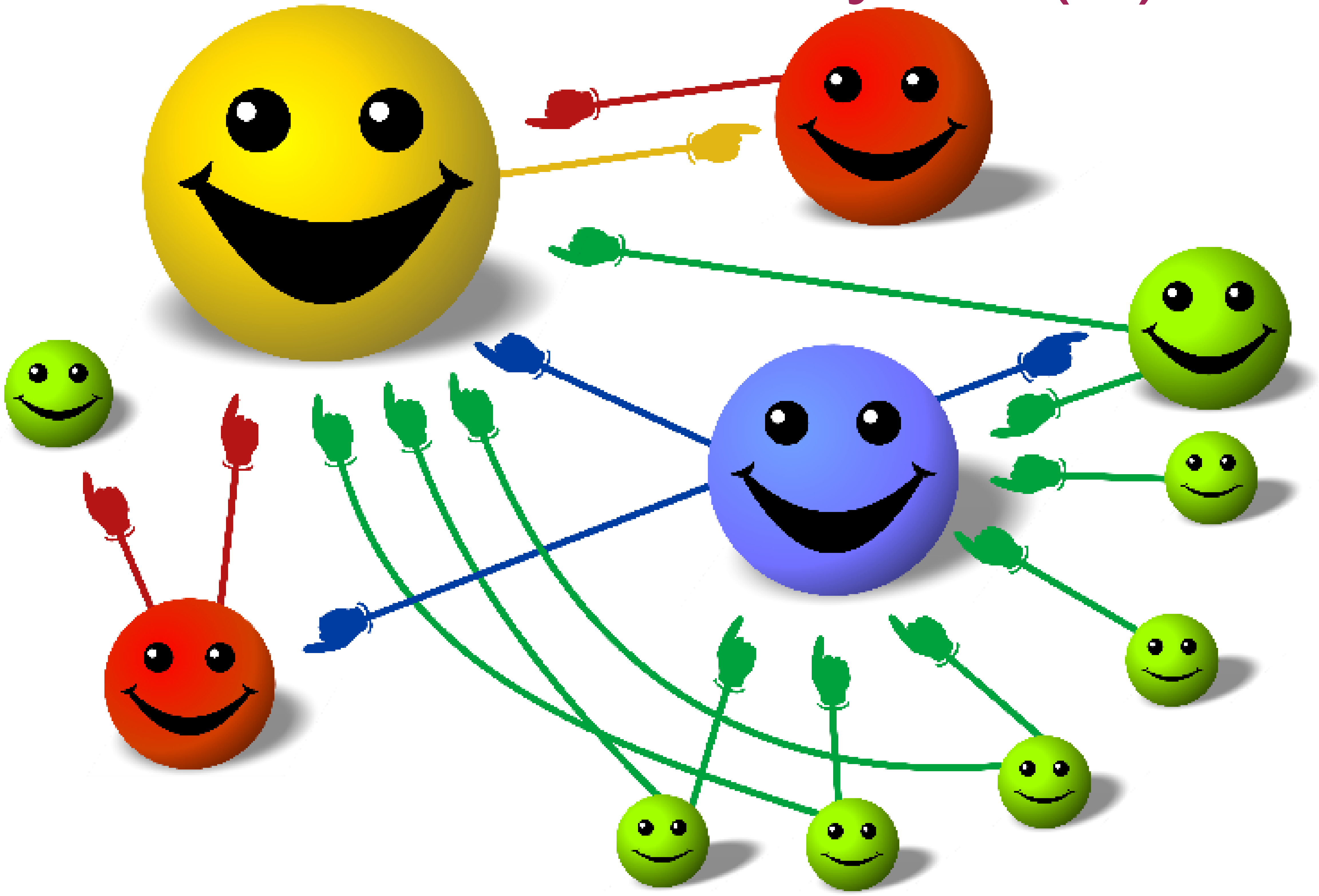
Page Rank Algorithm

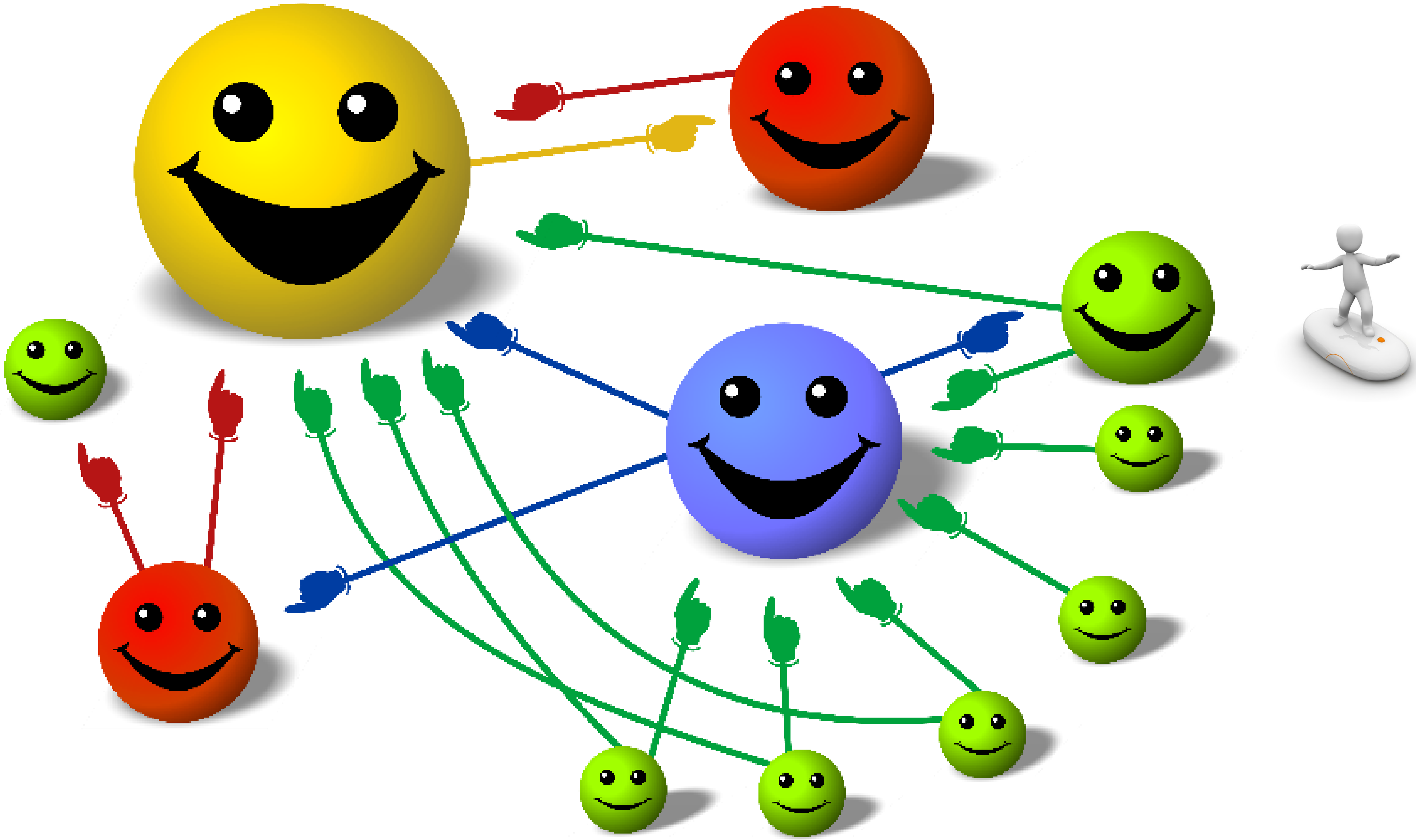
Google

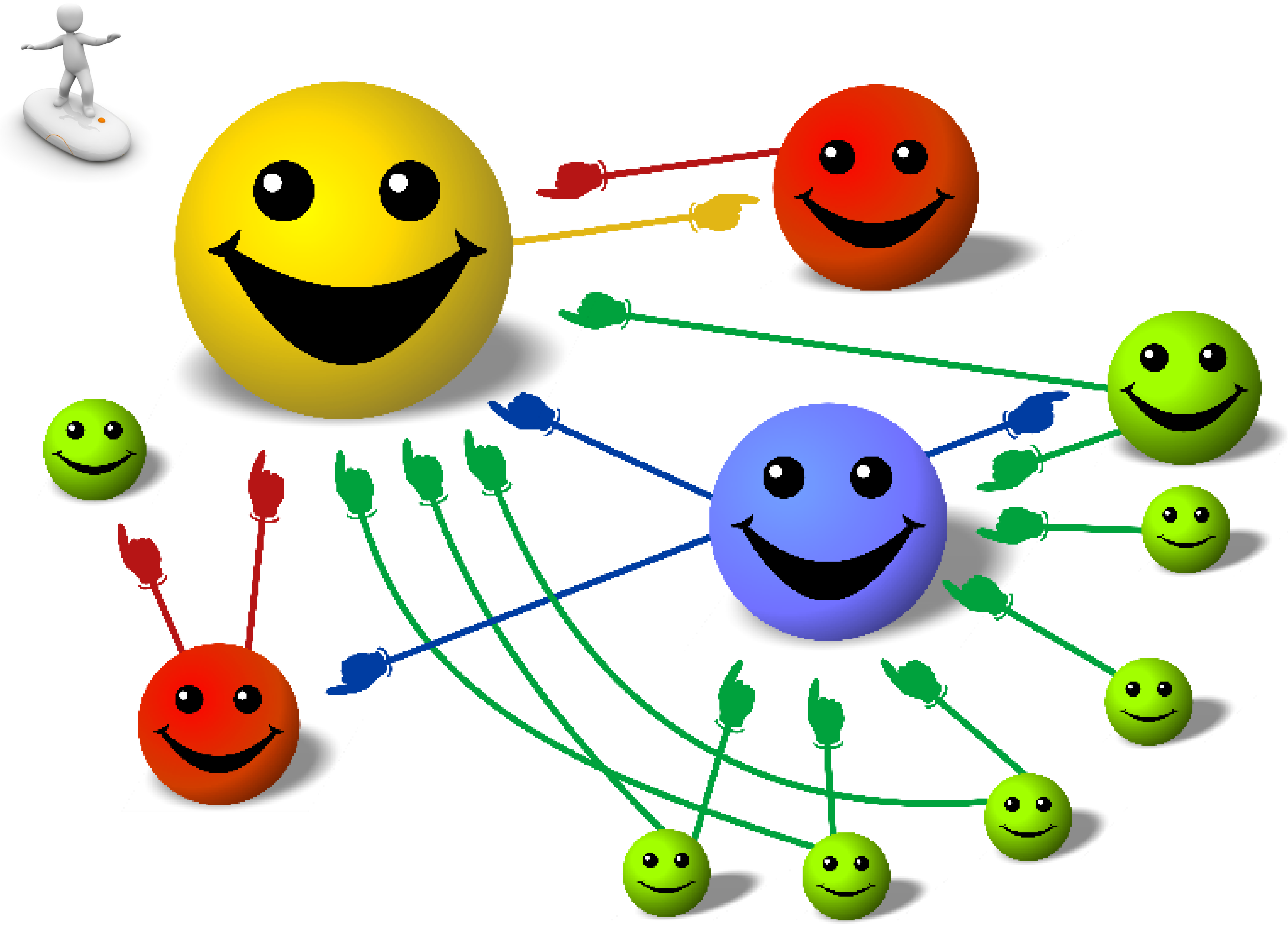
PageRank of E denoted by PR(E)

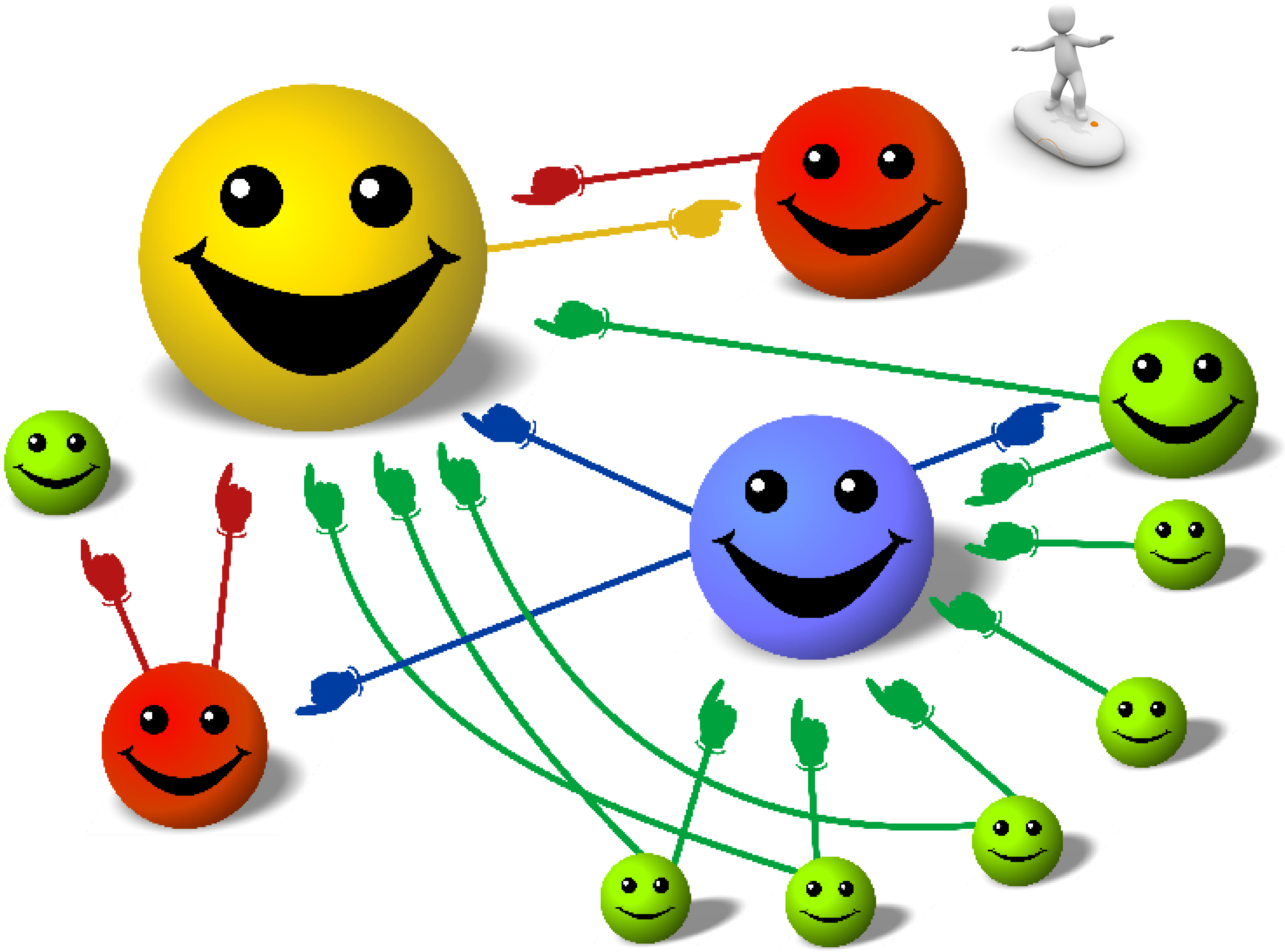
PageRank of E denoted by PR(E)

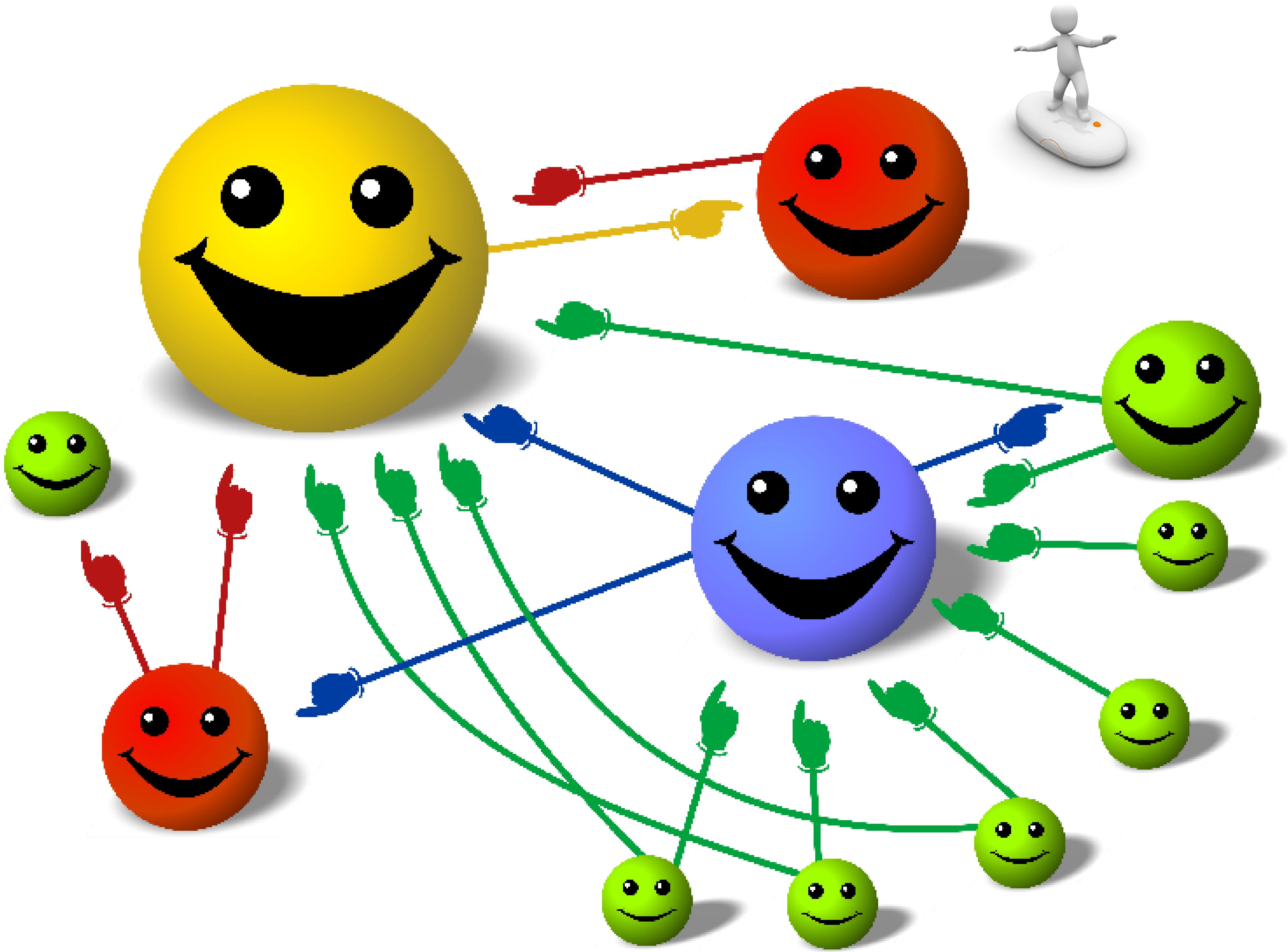



http://www
http://www

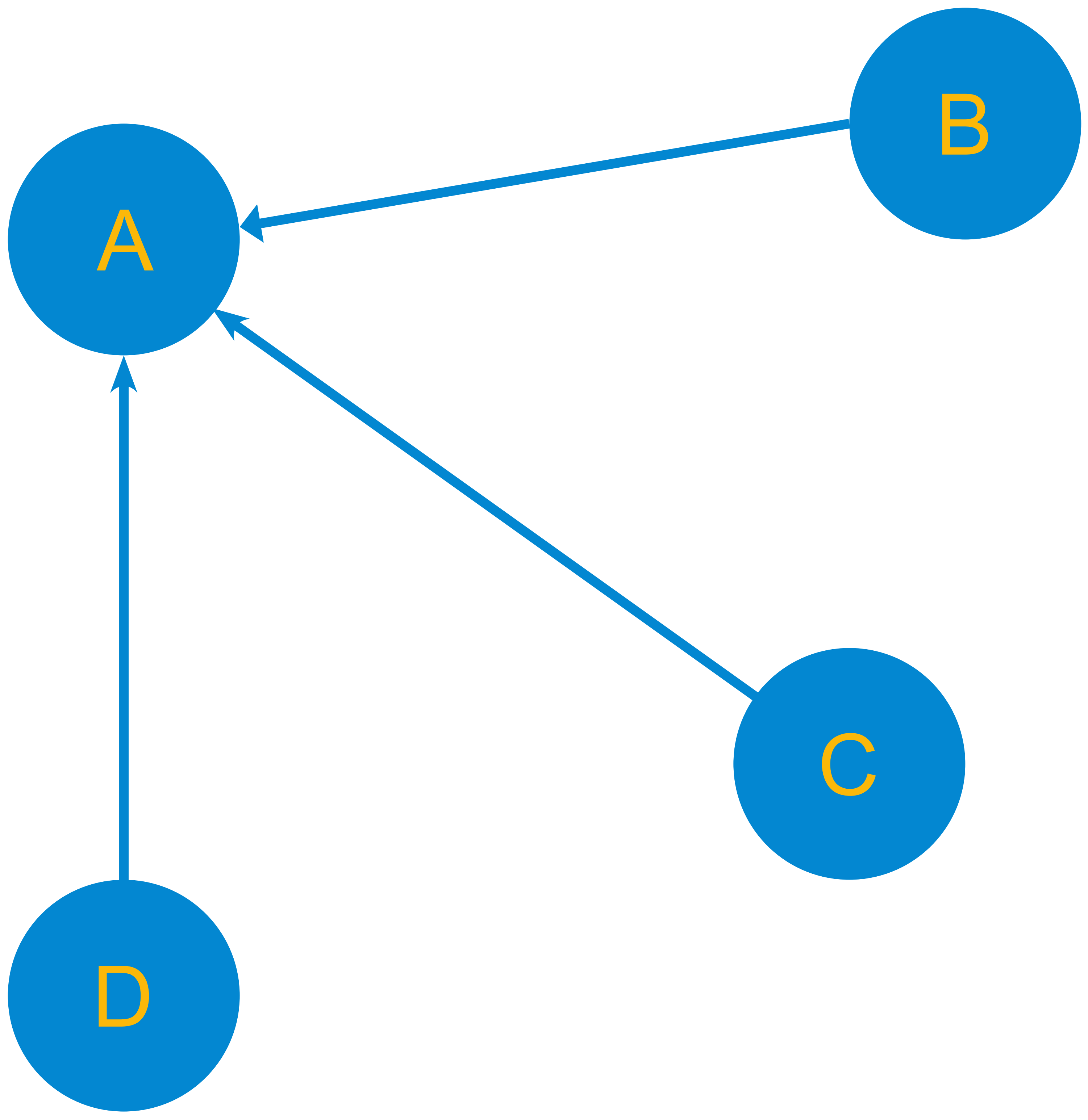


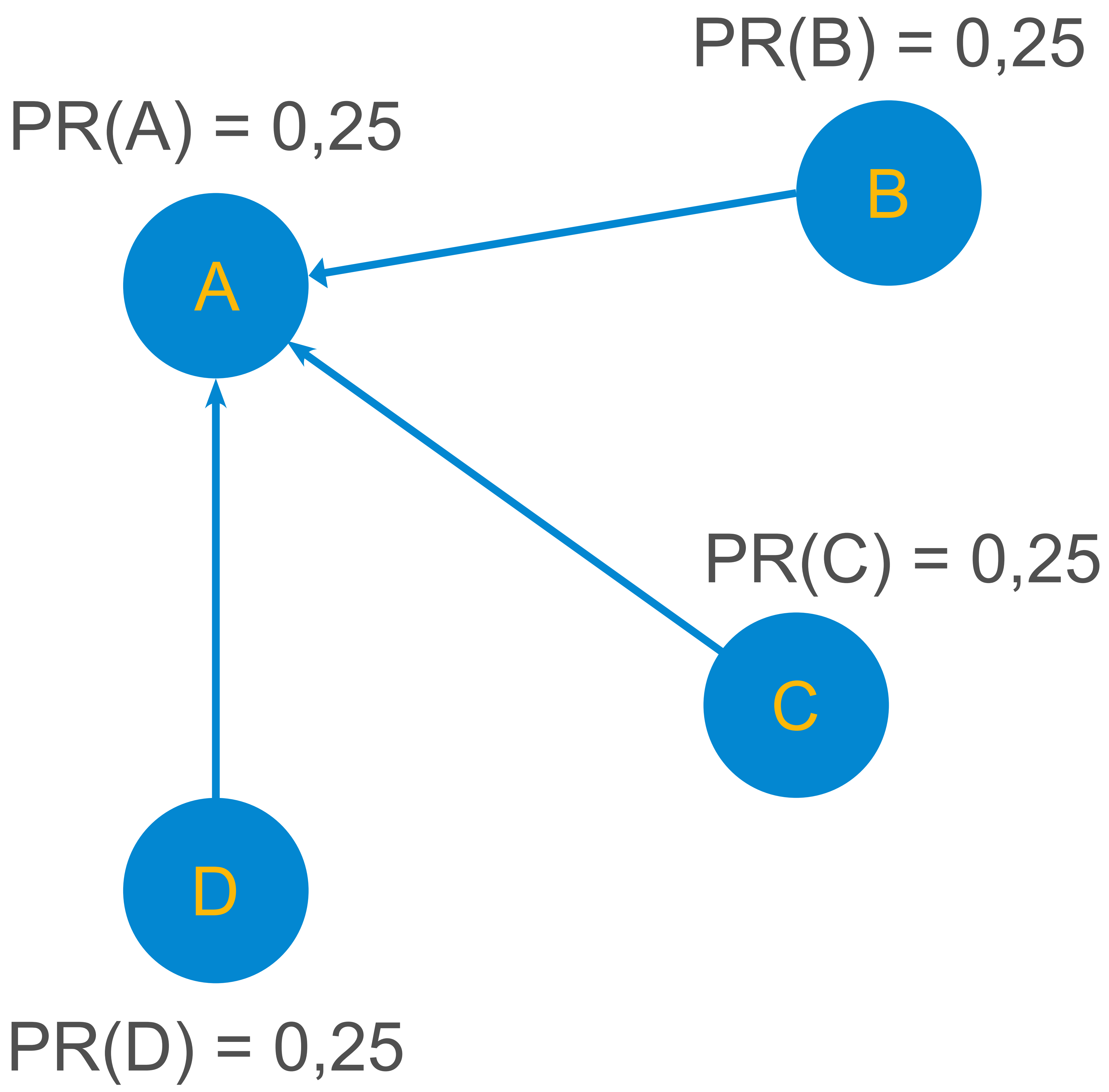


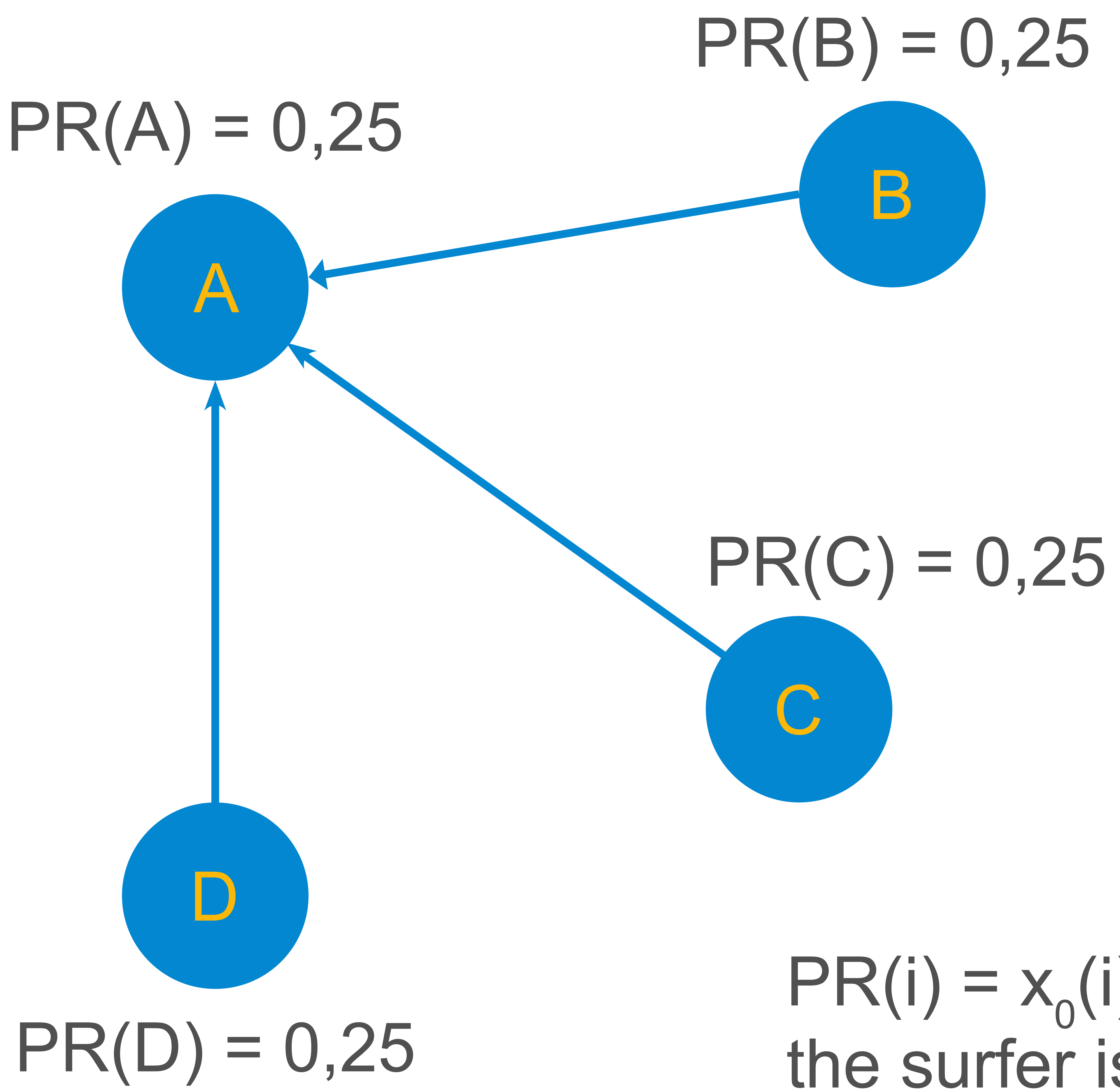




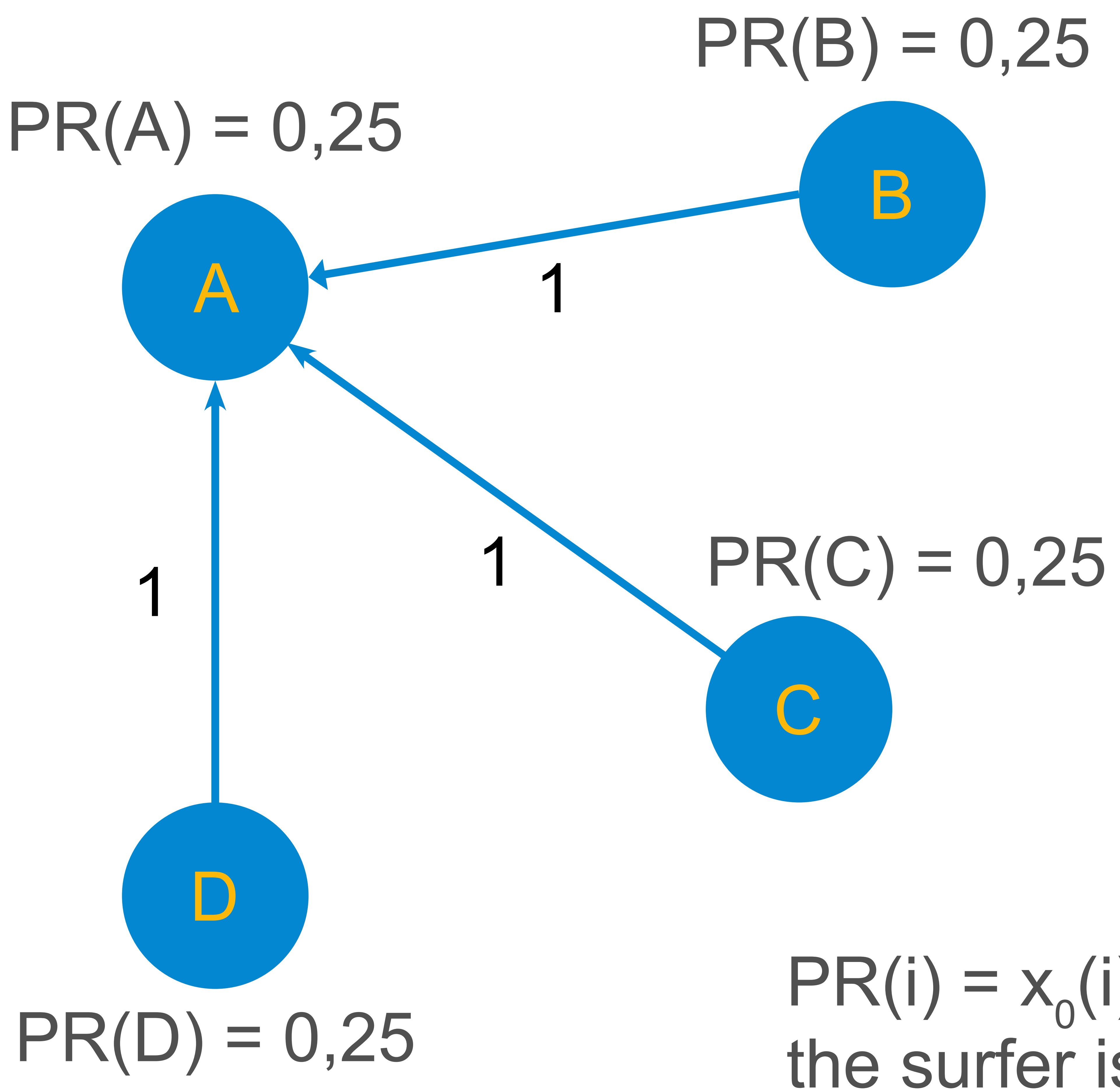
Stationary distribution $x^* = x^* P$


A
B
C
D

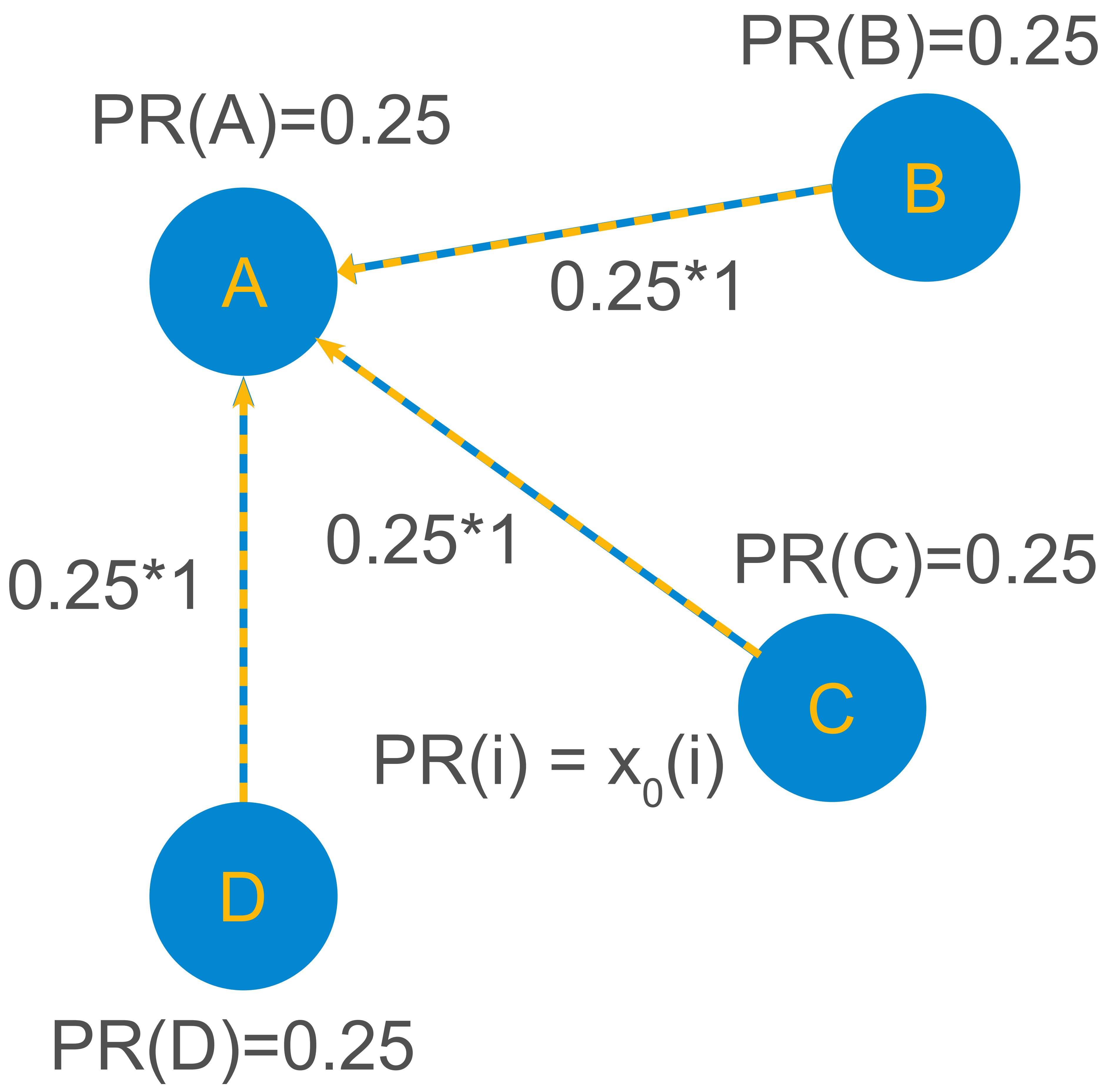

PR(B) = 0,25
PR(A) = 0,25
B
A
PR(C) = 0,25
C
D
PR(D) = 0,25



$PR(i) = x_0(i)$ = probability that the surfer is at node i at time 0

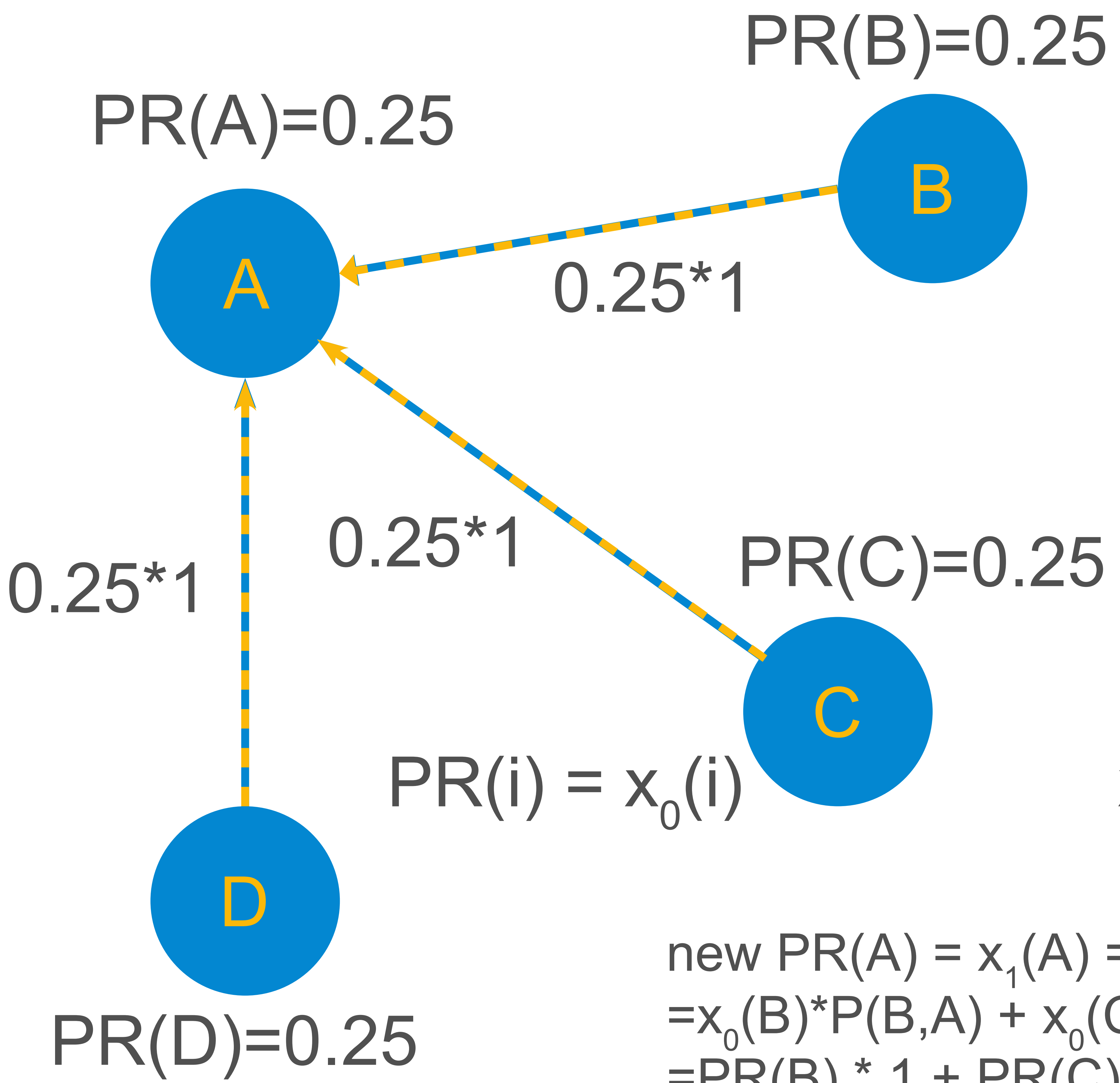


$PR(i) = x_0(i)$ = probability that the surfer is at node i at time 0



$$PR(i) = x_0(i)$$

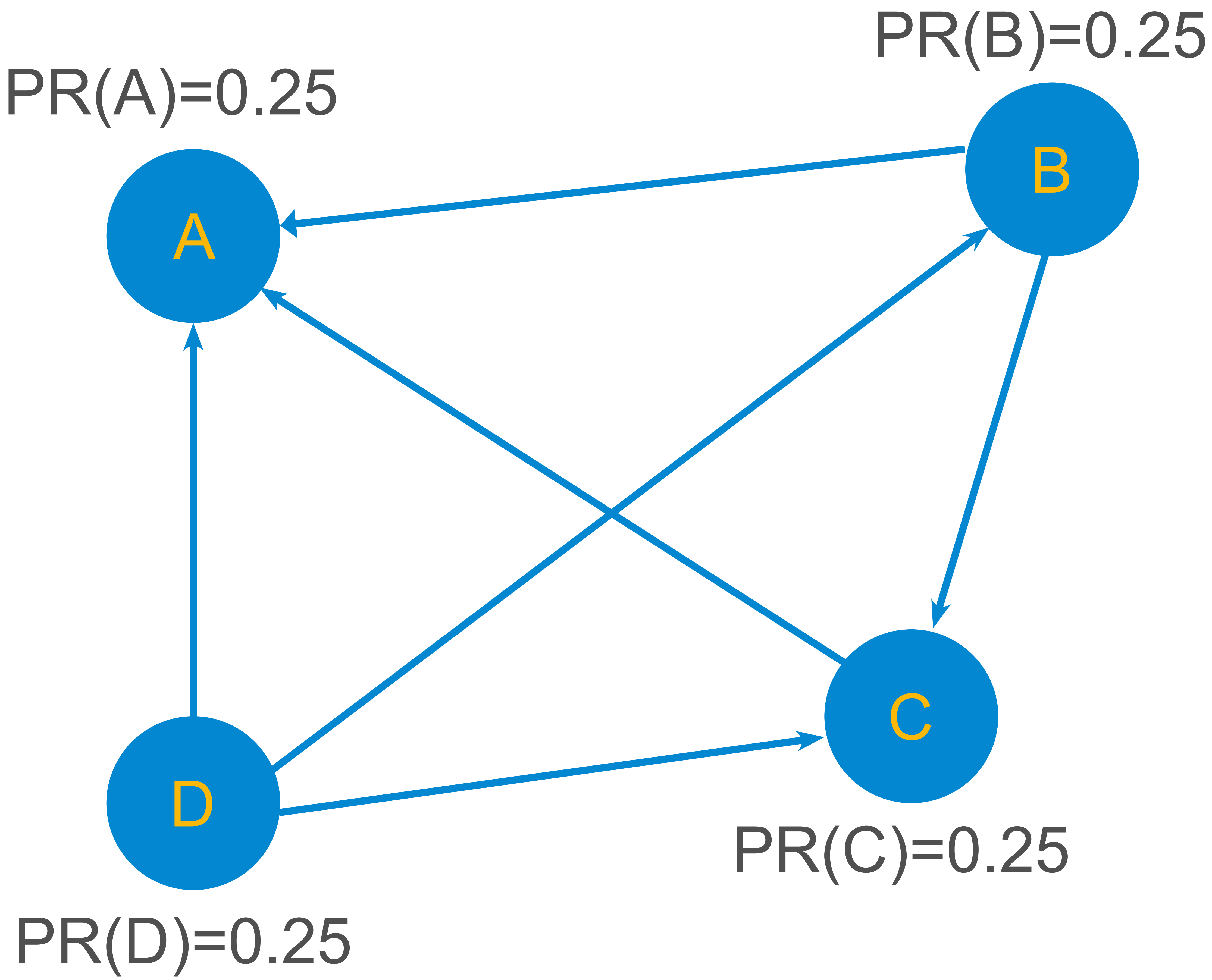
$$x_{t+1}(i) = \sum_j x_t(j)*P(j,i)$$

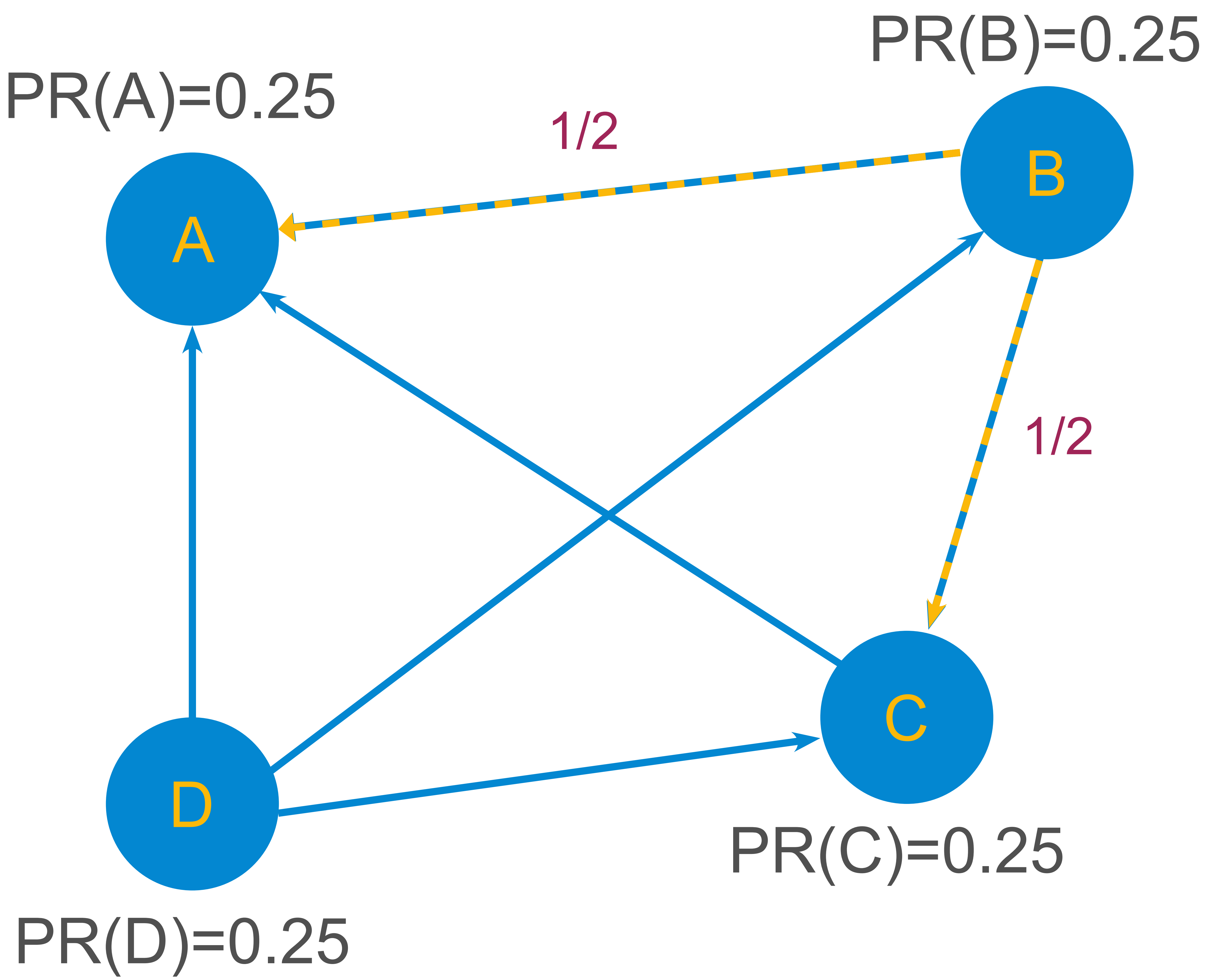


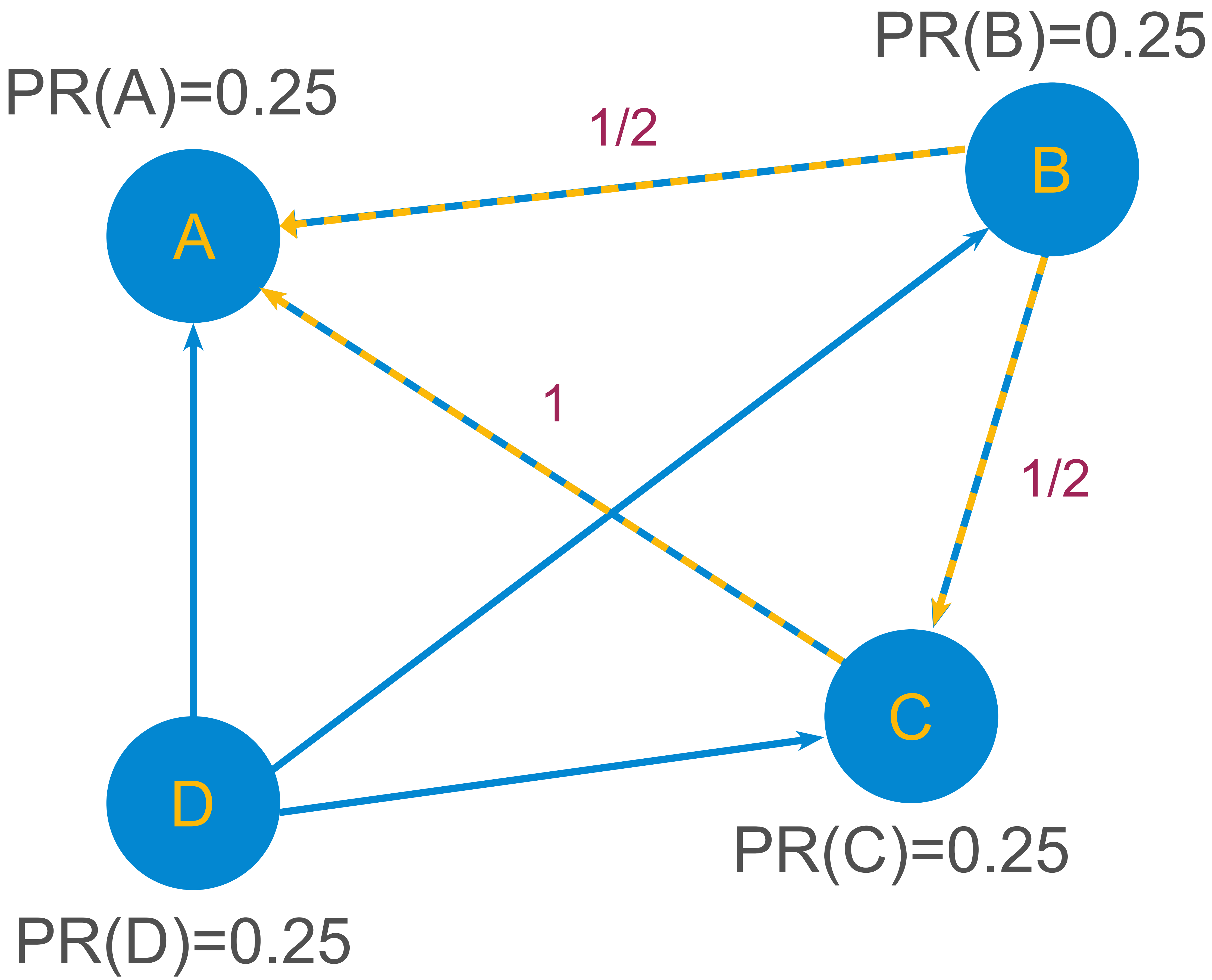
$PR(i) = x_0(i)$

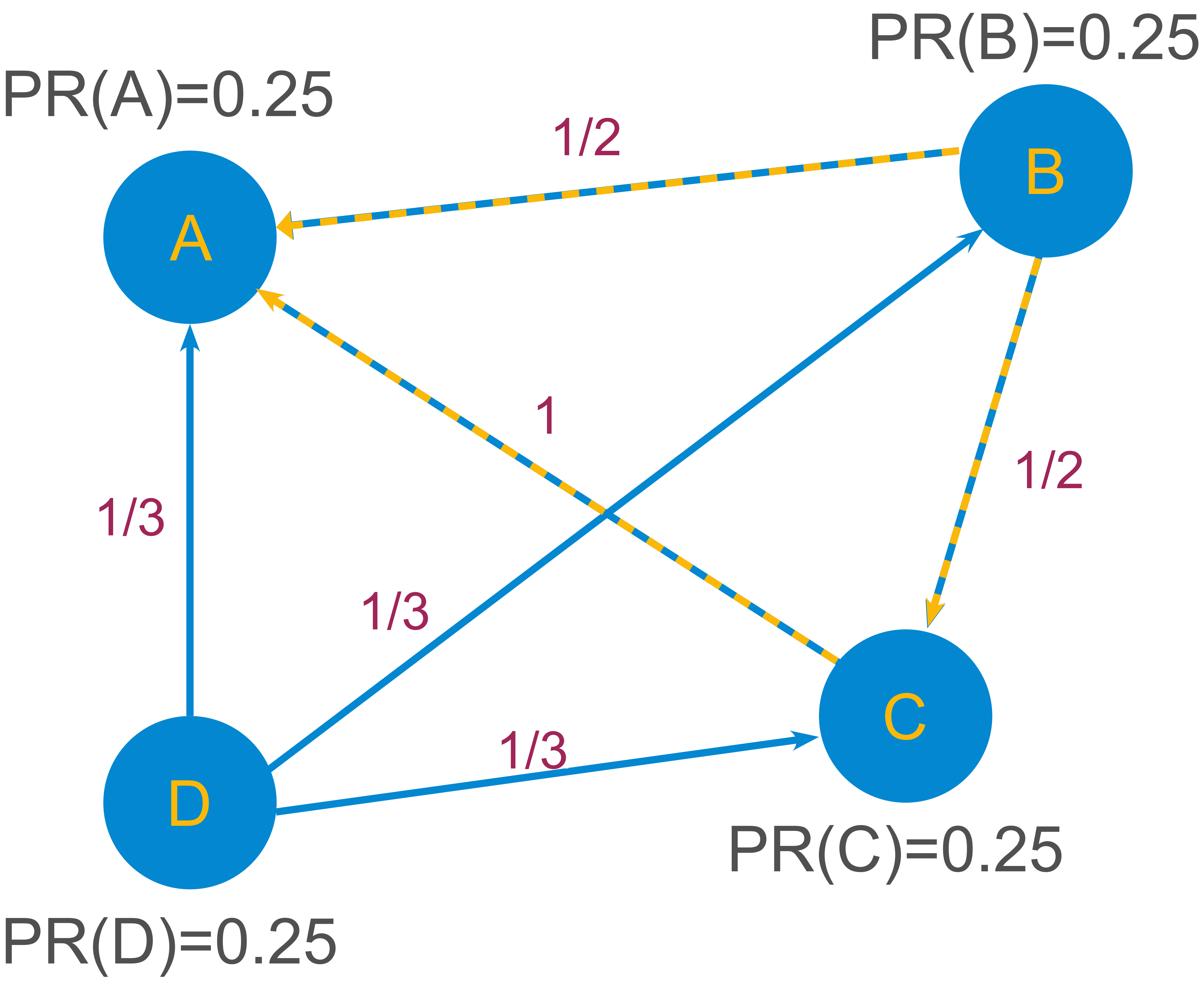
$x_{t+1}(i) = \sum_j x_t(j) * P(j,i)$

new $PR(A) = x_1(A) = \sum_j x_0(j) * P(j,A) =$
$= x_0(B) * P(B,A) + x_0(C) * P(C,A) + x_0(D) * P(D,A) =$
$= PR(B) * 1 + PR(C) * 1 + PR(D) * 1 = 0.75$


PR(B)=0.25
PR(A)=0.25
B
A
C
D
PR(C)=0.25
PR(D)=0.25


PR(B)=0.25
PR(A)=0.25
1/2
B
A
1/2
C
D
PR(C)=0.25
PR(D)=0.25


PR(A)=0.25
PR(B)=0.25
PR(C)=0.25
PR(D)=0.25
A
B
C
D
1/2
1
1/2



new PR(A) = x_1(A)=

=$\sum_j$ x_0(j)*P(j,A) =

= x_0(B)*P(B,A) +

+x_0(C)*P(C,A) +

+x_0(D)*P(D,A) =

= PR(B) * 1/2+

+ PR(C) * 1 +

+PR(D) * 1/3 =

=0.458

$L(v)$ - the number of vertex v outbound links

$L(v)$ - the number of vertex v outbound links

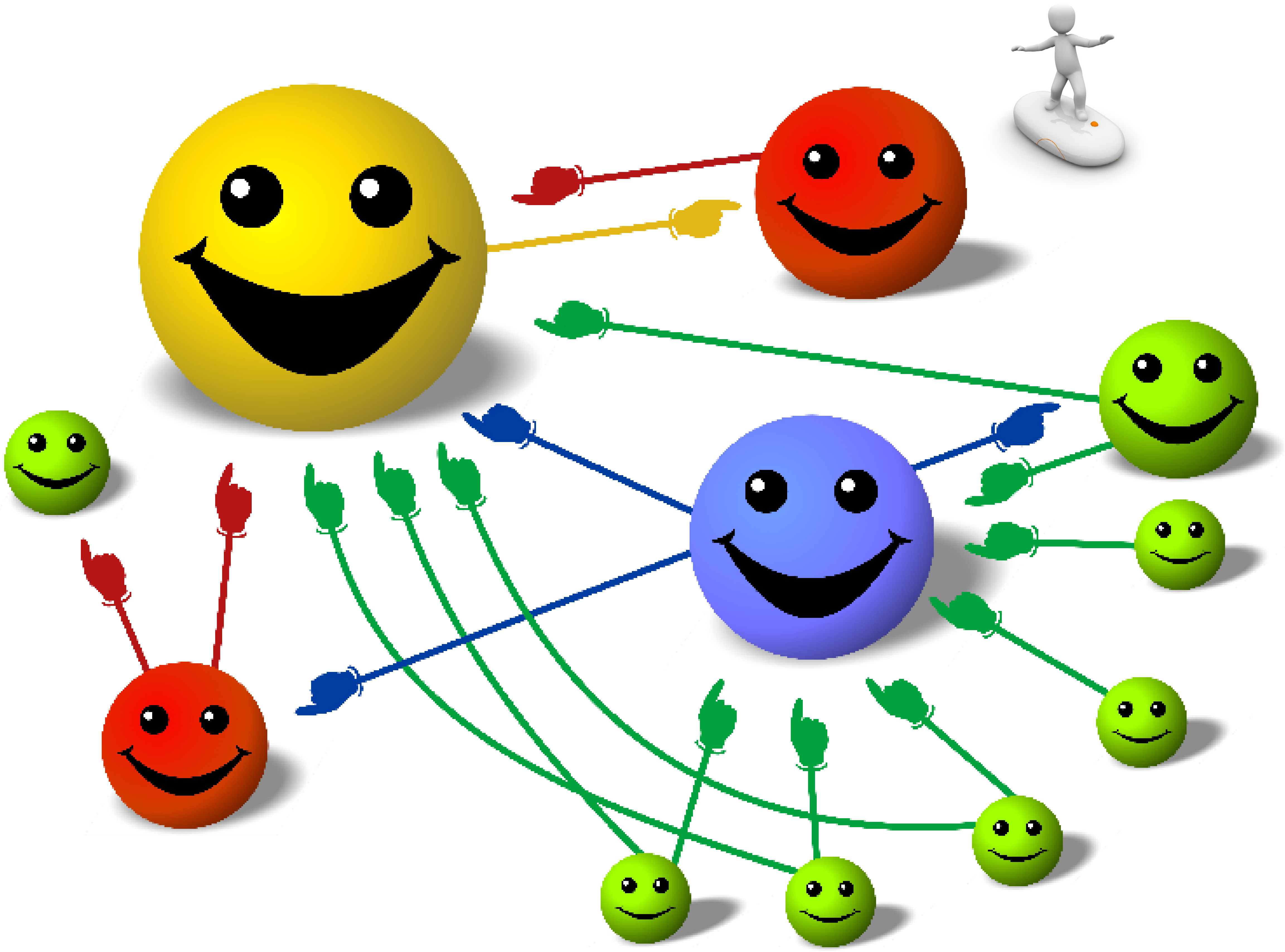
$\Gamma(v)$ - the set containing all pages linking to page v

L(v) - the number of vertex v outbound links

Γ(v) - the set containing all pages linking to page v

$$PR(u) = \sum_{v \in \Gamma(u)} \frac{PR(v)}{L(v)}$$


http://www
http://www



Stationary distribution $x^* = x^* P$

Stationary distribution

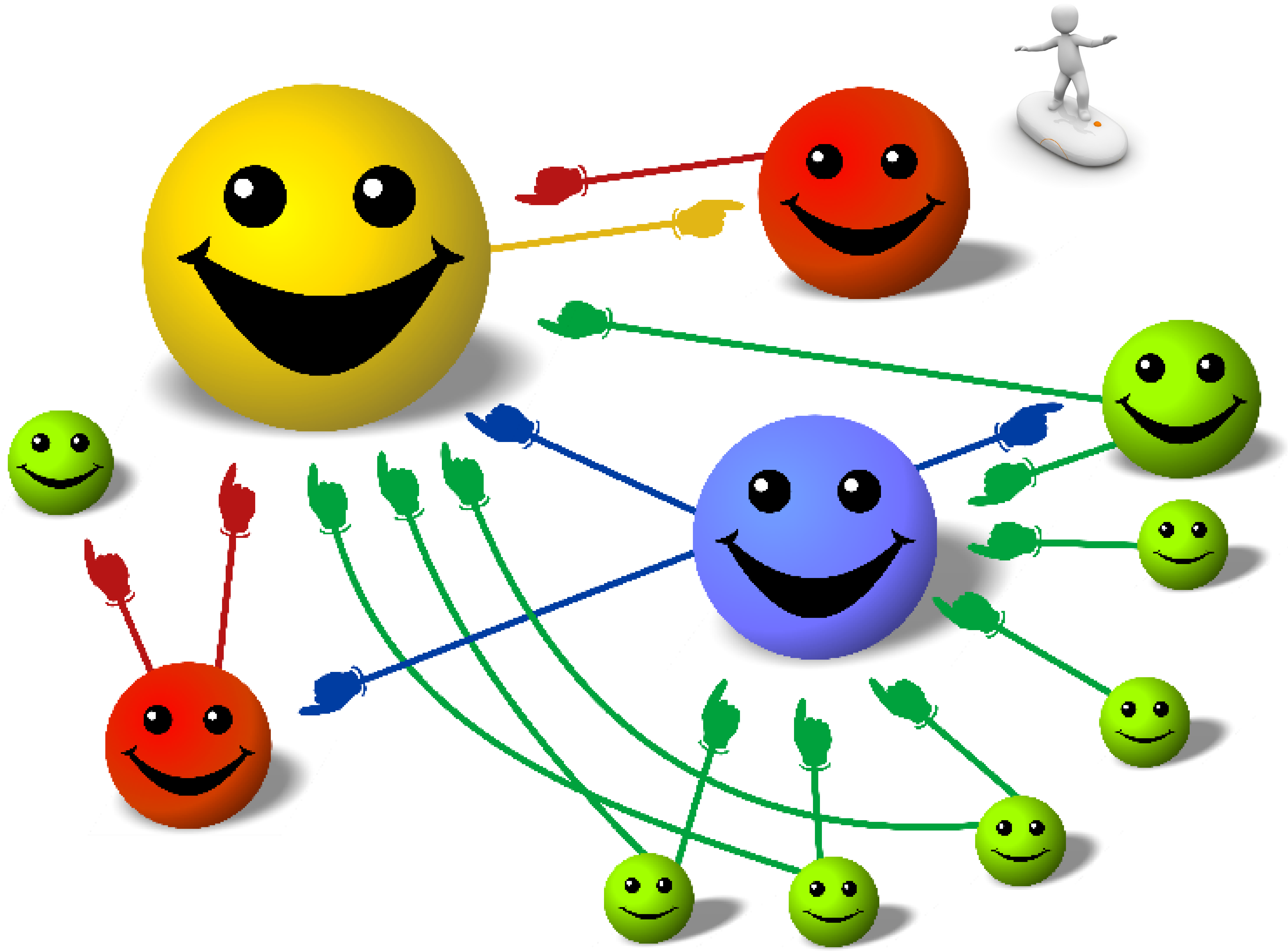
$x^* = x^* P$

Theorem

asserting that if a stochastic graph satisfies two conditions:

1. there is a path from every node to every node
2. the greatest common divider of all the cycle lengths id 1

then there is a unique stationary probability distribution



Stationary distribution $x^* = x^* P$

The probability, at any step, that the person will continue surfing is a damping factor d

$d = 0.85$

$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \cdots$$

$$PR(A) = \frac{1-d}{N} + d\left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \cdots\right)$$

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in \Gamma(p_i)} \frac{PR(p_j)}{L(p_j)}$$

where $p_1, p_{2,} ..., p_N$ are the pages under consideration

$\Gamma(p_i)$ is the set of pages that link p_i

$L(p_i)$ is the number of outbound links on page p_i

and N is the total number of pages

At t= 0, an initial probability distribution is assumed, usually $PR(p_i;0) = \frac{1}{N}$

At each time step $PR(p_i;t+1) = \frac{1-d}{N} + d \sum_{p_j \in \Gamma(p_i)} \frac{PR(p_j;t)}{L(p_j)}$

At t= 0, an initial probability distribution is assumed, usually $PR(p_i;0) = \frac{1}{N}$

At each time step $PR(p_i;t+1) = \frac{1-d}{N} + d \sum_{p_j \in \Gamma(p_i)} \frac{PR(p_j;t)}{L(p_j)}$

The computation ends:

1. After fixed number of iterations

At t= 0, an initial probability distribution is assumed, usually $PR(p_i;0) = \frac{1}{N}$

At each time step $PR(p_i;t+1) = \frac{1-d}{N} + d \sum_{p_j \in \Gamma(p_i)} \frac{PR(p_j;t)}{L(p_j)}$

The computation ends:

1. After fixed number of iterations
2. When convergence is assumed

$$\left(\sum_{i=0}^{N} |PR(p_i, t+1) - PR(p_i, t)| \right) < \epsilon$$

At t= 0, an initial probability distribution is assumed, usually PR(p_i;0) $= \frac{1}{N}$

At each time step PR(p_i;t+1) = $\frac{1-d}{N} + d \sum_{p_j \in \Gamma(p_i)} \frac{PR(p_j;t)}{L(p_j)}$

The computation ends:

1. After fixed number of iterations
2. When convergence is assumed

$$\left(\sum_{i=0}^{N} |PR(p_i, t+1) - PR(p_i, t)| \right) < \epsilon$$

$$PR(p_i; t+1) \xrightarrow[t \to \infty]{} x^*(p_i)$$

Summary

- You have known what is **PageRank (PR)** - a first algorithm by Google Search to rank websites in their search engine results.

Summary

- You have known what is **PageRank (PR)** - a first algorithm by Google Search to rank websites in their search engine results.
- You have learned how to calculate iteratively PageRank for every vertex in our graph.