Non-Standard-Datenbanken

Approximative Analyse von Graphstrukturen

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Acknowledgments

Quite a few slides (indicated) have been taken from a lectures on Leveraging Big Data (2013/14) by Edith Cohen at Tel-Aviv-University

Presentations are possibly adapted and extended

Faults are mine

Graph Datasets

- Hyperlinks (the Web)
- Social graphs (Facebook, Twitter, LinkedIn,…)
- Communication networks
- Protein interaction networks
- …

Properties of graphs

- Snapshot or with time dimension (dynamic)
- One or more types of abstract entities (node labels: people, pages, products) possibly with named attributes (integers, reals. …)[1]
- One or more types of edges
(edge labels: has-informed, can-communicate-with, …)
- Directed/undirected edges

[1] RDF graphs try to be more uniform and use specific edges with label rdf:type for identifying node labels and also use edges as attributes

Recap: Mining the link structure

Network- and node-level properties

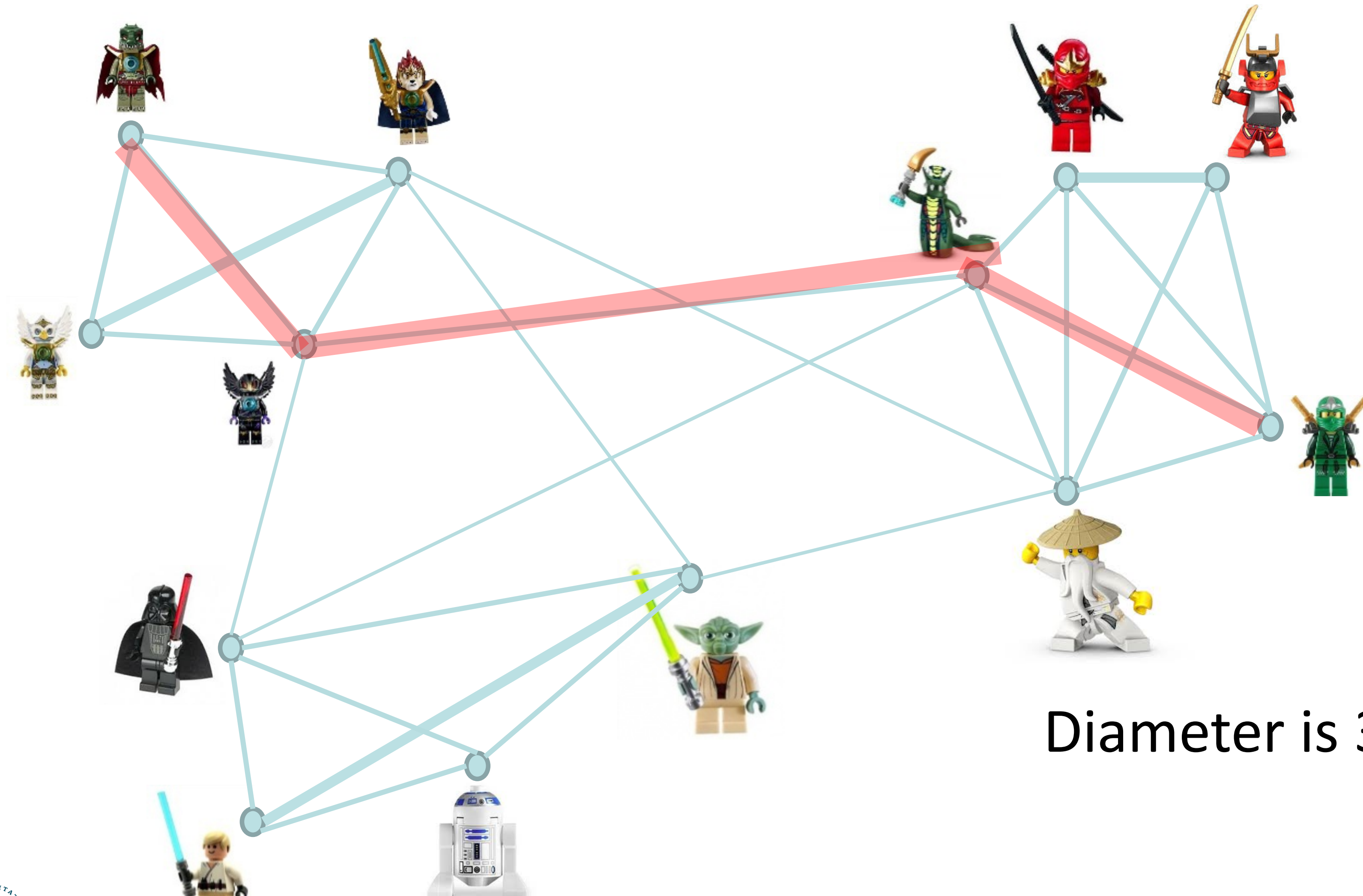
- Similarity of nodes (e.g., distance distribution, reachability size)
  - Link prediction, targeted ads, friend/product recommendations, …)
- Centrality (e.g., betweenness w.r.t. all-pairs shortest paths)
  - Importance of nodes

$$C_B(n_i) = \sum_{j<k} g_{jk}(n_i) / g_{jk}$$

g_{jk} = number of minimal paths between nodes j and k
$g_{jk}(n)$ = number of minimal paths between nodes j and k that contain n

- Diameter (longest shortest s-t path)
  - Connectedness of the network overall

Diameter (longest shortest path between two nodes)

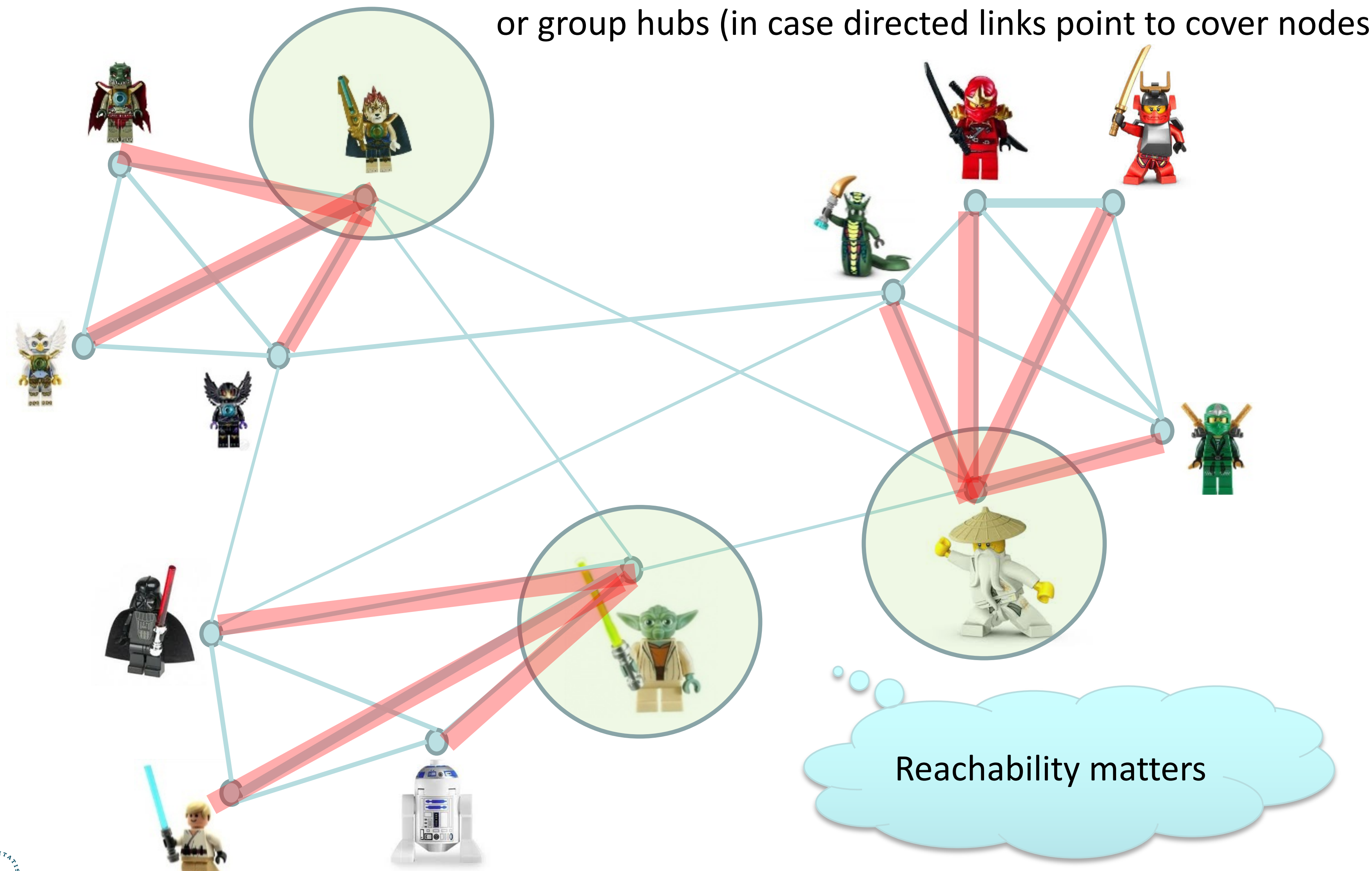


Diameter is 3

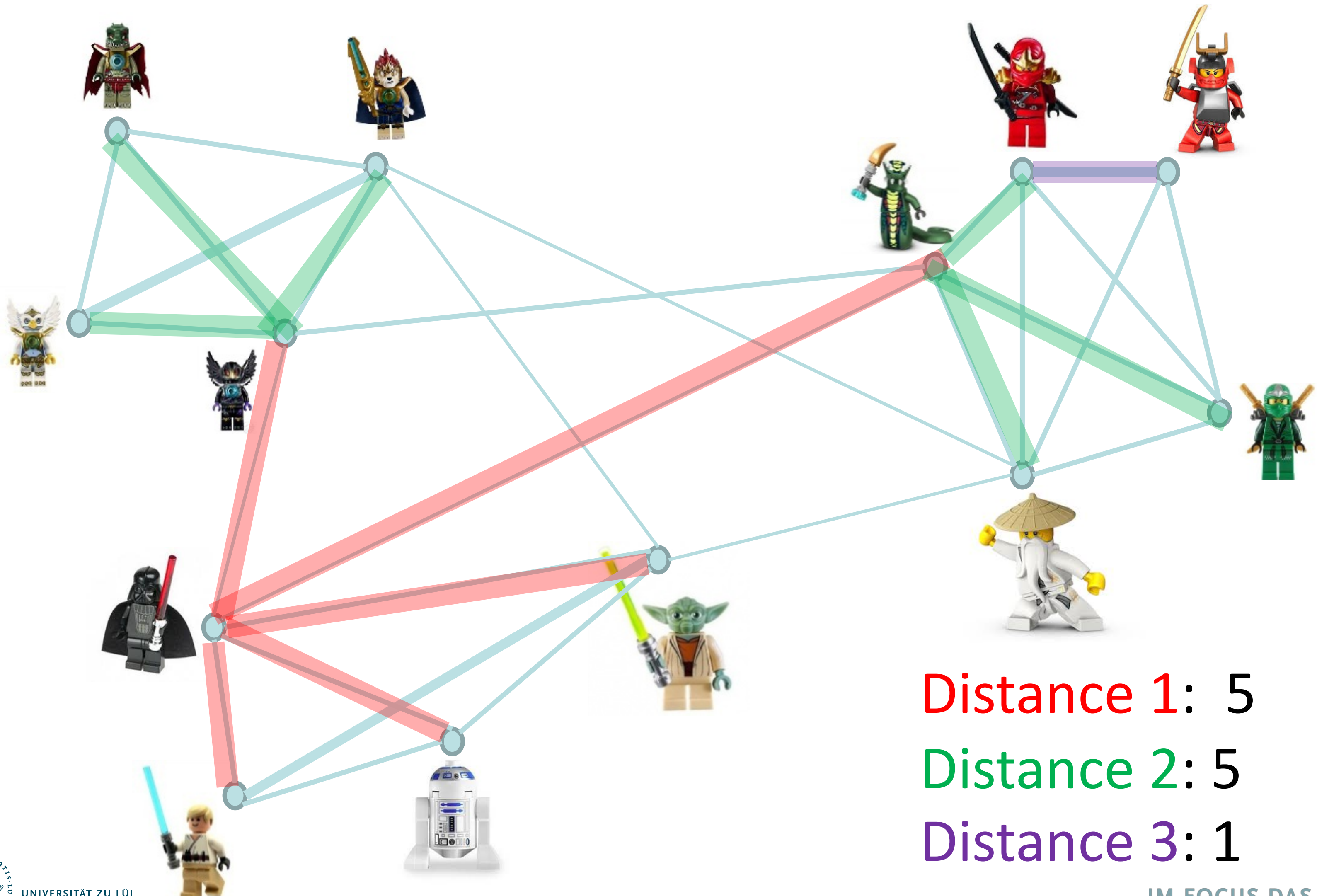
Computing Diameter

- Can we run all pairs shortest path algorithms on large graphs with non-negative edge labels?
  - $O(n \cdot T_{Dijkstra}(n,m)) = O(n(n \log n + m))$ with Fibonacci heaps
- Hardly!

Cover (undirected)



Distance distribution of 



Distance 1: 5

Distance 2: 5

Distance 3: 1

Algorithm Design Principles for Big Data

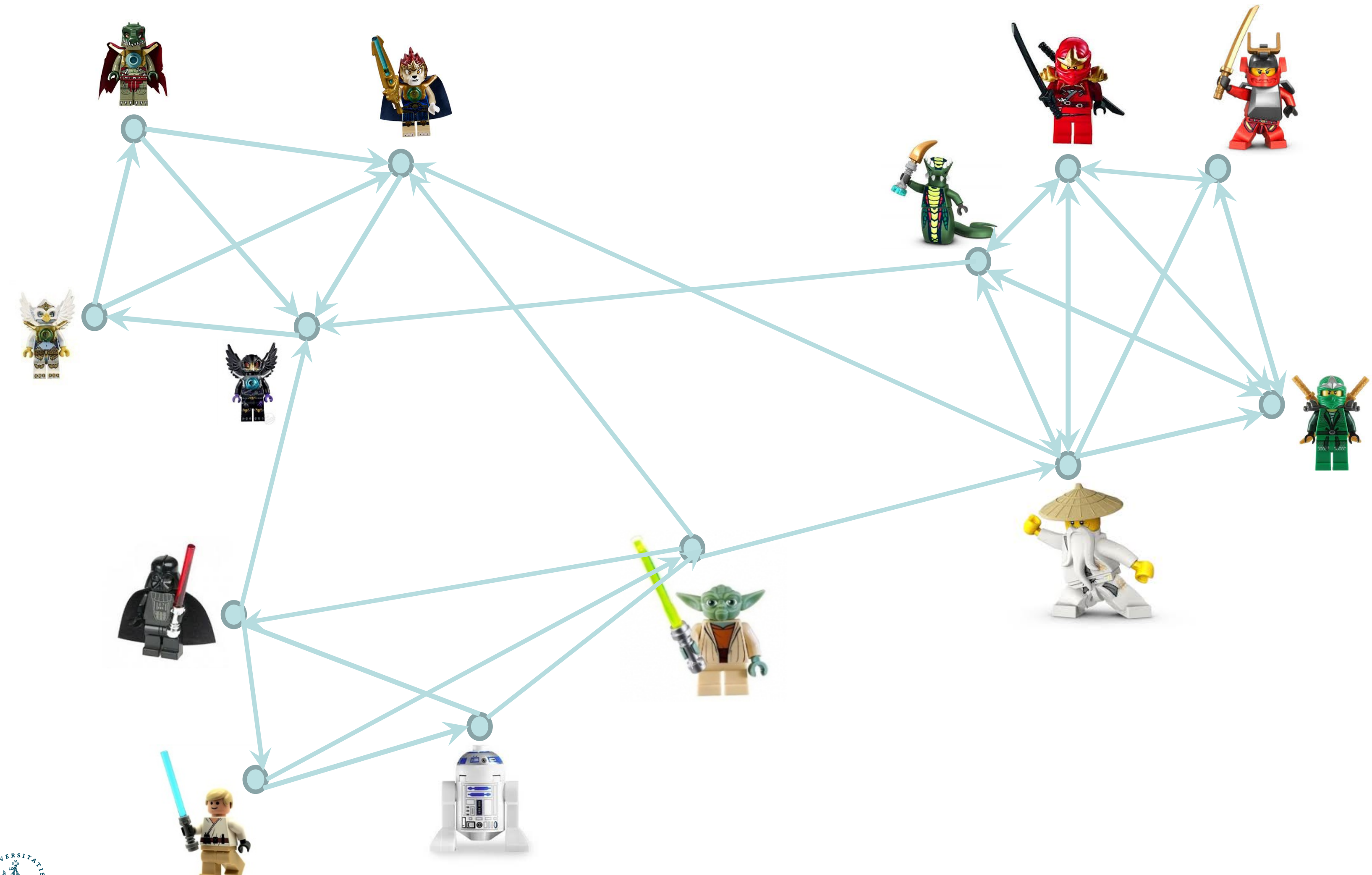
- Settle for approximations
- Keep memory polylog in data size
- Keep total computation/communication "linear" in the size of the data
- Parallelize (minimize chains of dependencies)

Node Sketches

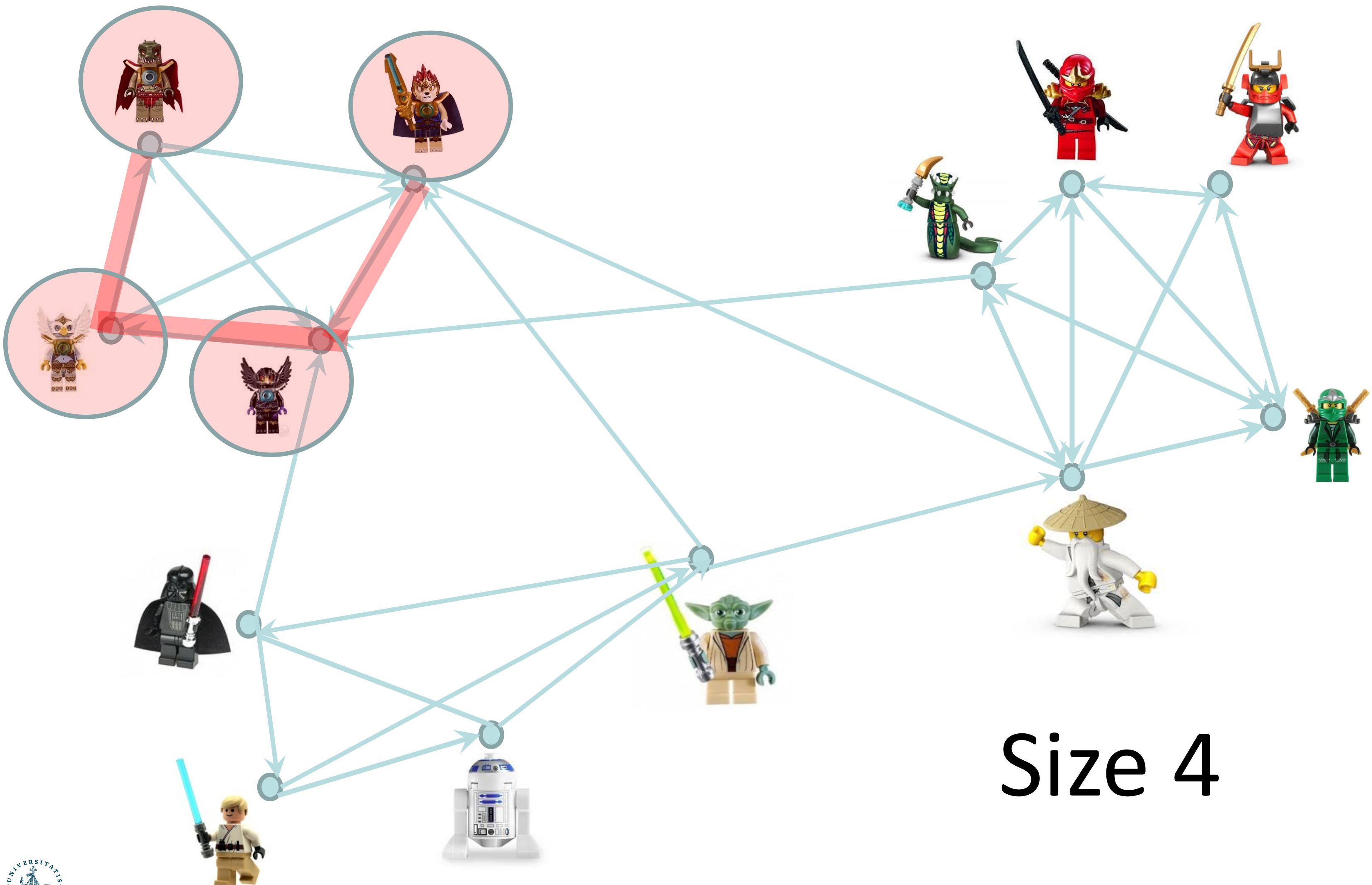
Sketching:

- Compute a sketch for each node, <u>efficiently</u>
- From sketch(es) one can estimate properties that are "harder" to compute exactly
  - Min-Hash sketches of reachability sets
    - Reachability estimated
    - Do not try shortest path algorithms on two nodes for which there is no reachability relation
  - All-distances sketches (ADS)
    - Betweenness centrality estimated

Sketching Reachability Sets

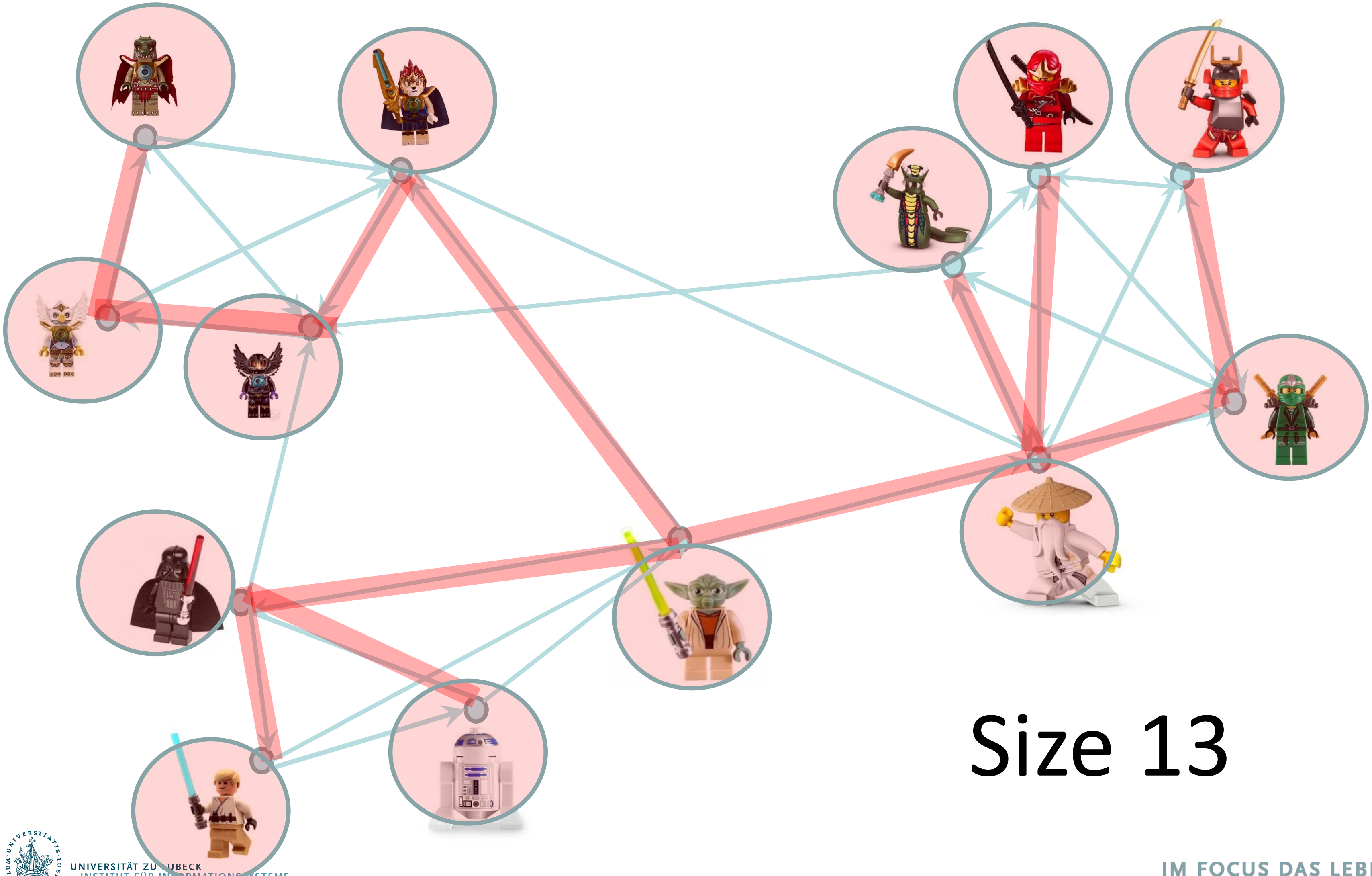


Reachability Set of 



Size 4

Reachability Set of 



Size 13

Why sketch reachability sets ?

From reachability sketch(es) we can:

- Estimate cardinality of reachability set
- Get a sample of the reachable nodes
- Estimate relations between reachability sets (e.g., Jaccard similarity)

➢ Exact **computation** is *costly*: $O(mn)$ with n nodes and m edges, **representation size** is massive: does not scale to large networks!

- Min-Hashing comes to the rescue

Edith Cohen and Haim Kaplan. Summarizing data using bottom-k sketches.
In Proc. Symposium on Principles of distributed computing (PODC '07). ACM, pp. 225-234, 2007

Recap: Min-Hashing

- Gegeben zwei Mengen A und B von Objekten
- Zum Beispiel sind A und B Text-Dokumente mit Wörtern.
- Ein oft benutztes Ähnlichkeitsmaß ist der **Jaccard Koeffizient**:

$$J(A,B) := \frac{|A \cap B|}{|A \cup B|}$$

Siehe Teil Information Retrieval in dieser Vorlesung

Idee hinter Min-Hashing

- Berechne für alle Elemente einer Menge A eine Hashfunktion, die $a \in A$ einem Integer-Wert zuweist.

Siehe Wörterbucher in AuD

- Betrachte den kleinsten dieser Werte $h_{min}(A)$
- Wie groß ist die Wahrscheinlichkeit, dass zwei Mengen A und B dieser min-Wert identisch ist?

Andrei Broder, On the resemblance and containment of documents. In SEQUENCES '97 Proceedings of the Compression and Complexity of Sequences, **1997**

Recap: Min-Hashing (2)

- Die Wahrscheinlichkeit $Pr[h_{min}(A) = h_{min}(B)]$ steht in direkter Verbindung zum Jaccard-Koeffizienten:

$$Pr[h_{min}(A) = h_{min}(B)] = \frac{|A \cap B|}{|A \cup B|}$$

Anwendung

- Suche nach ähnlichen Mengen: Betrachte nur Paare von Mengen, deren min-Wert identisch ist
- Bzw., nehme $Pr[h_{min}(A) = h_{min}(B)]$ als Näherung für den Jaccard-Koeffizienten
- Wie gut funktioniert das?

Recap: Min-Hashing - Mehrere min-Werte bzw. Hashfunktionen

Mehrere (k) Hash-Funktionen mit je einem min-Wert

- Betrachte k (unabhängige) Hashfunktionen, die jeweils einen min-Wert liefern.
- Approximiere $J(A,B)$ mit Anteil der übereinstimmenden min-Werte.
- Wie im Beispiel zuvor.

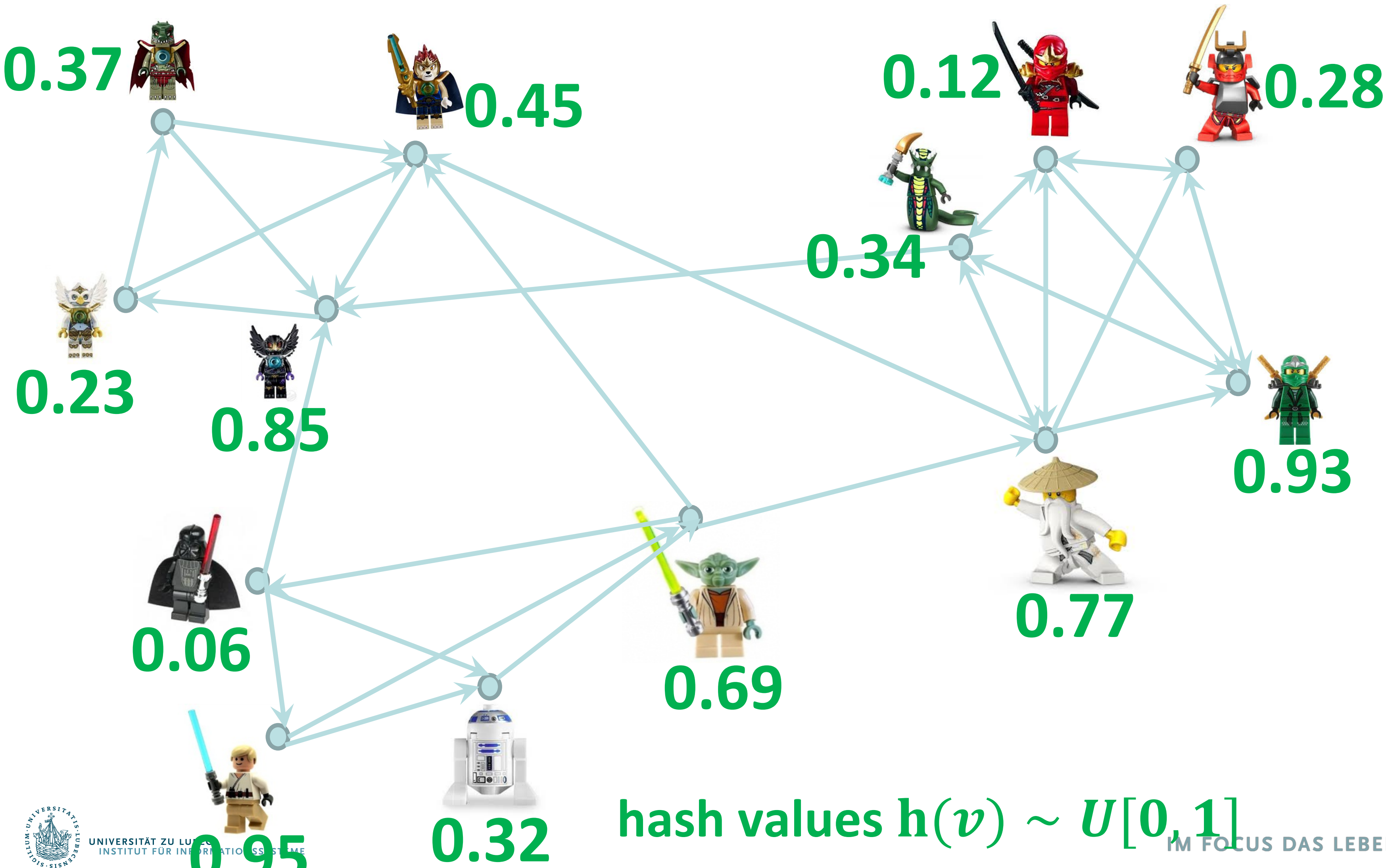
Mehrere (k) min-Werte & eine Hashfunktion

- Betrachte nur eine Hashfunktion, aber nehme von dieser die k kleinsten Werte.

Der Fehler ist bei bei

Nicht Top-k
sondern Bottom-k

Min-Hash sketches of all Reachability sets



hash values $\mathbf{h}(v) \sim U[0,1]$

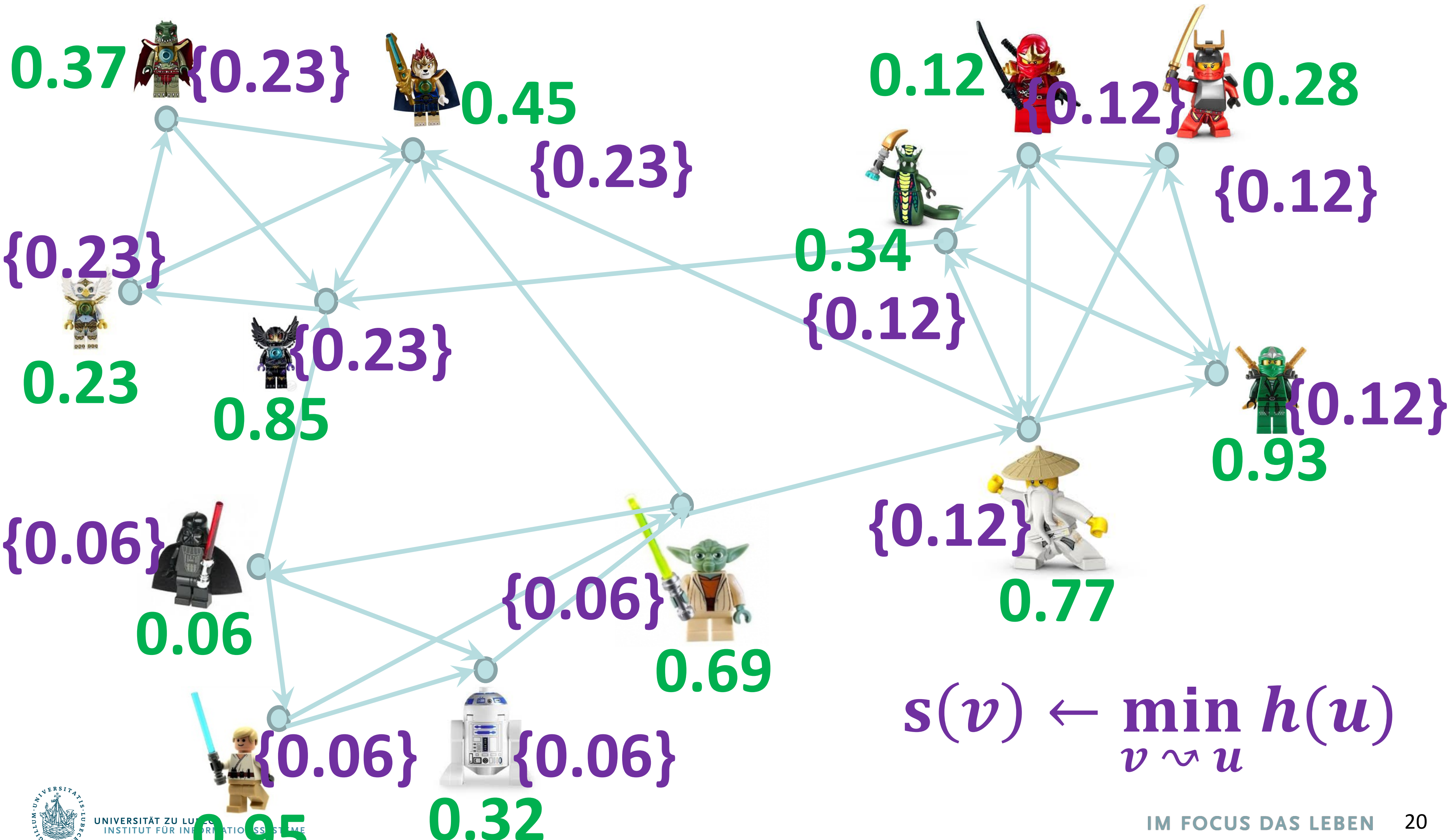
Min-Hash sketches of all Reachability Sets: $k = 1$

For each v: $\mathbf{s}(\boldsymbol{v}) \leftarrow \min_{\boldsymbol{v} \leadsto \boldsymbol{u}} \boldsymbol{h}(\boldsymbol{u})$

Depending on application, may also want to include node ID in sketch:

$$\operatorname*{argmin}_{\boldsymbol{v} \leadsto \boldsymbol{u}} \boldsymbol{h}(\boldsymbol{u})$$

Min-Hash sketches of all Reachability Sets: $k = 1$



$$s(v) \leftarrow \min_{v \rightsquigarrow u} h(u)$$

Communities and Reachability

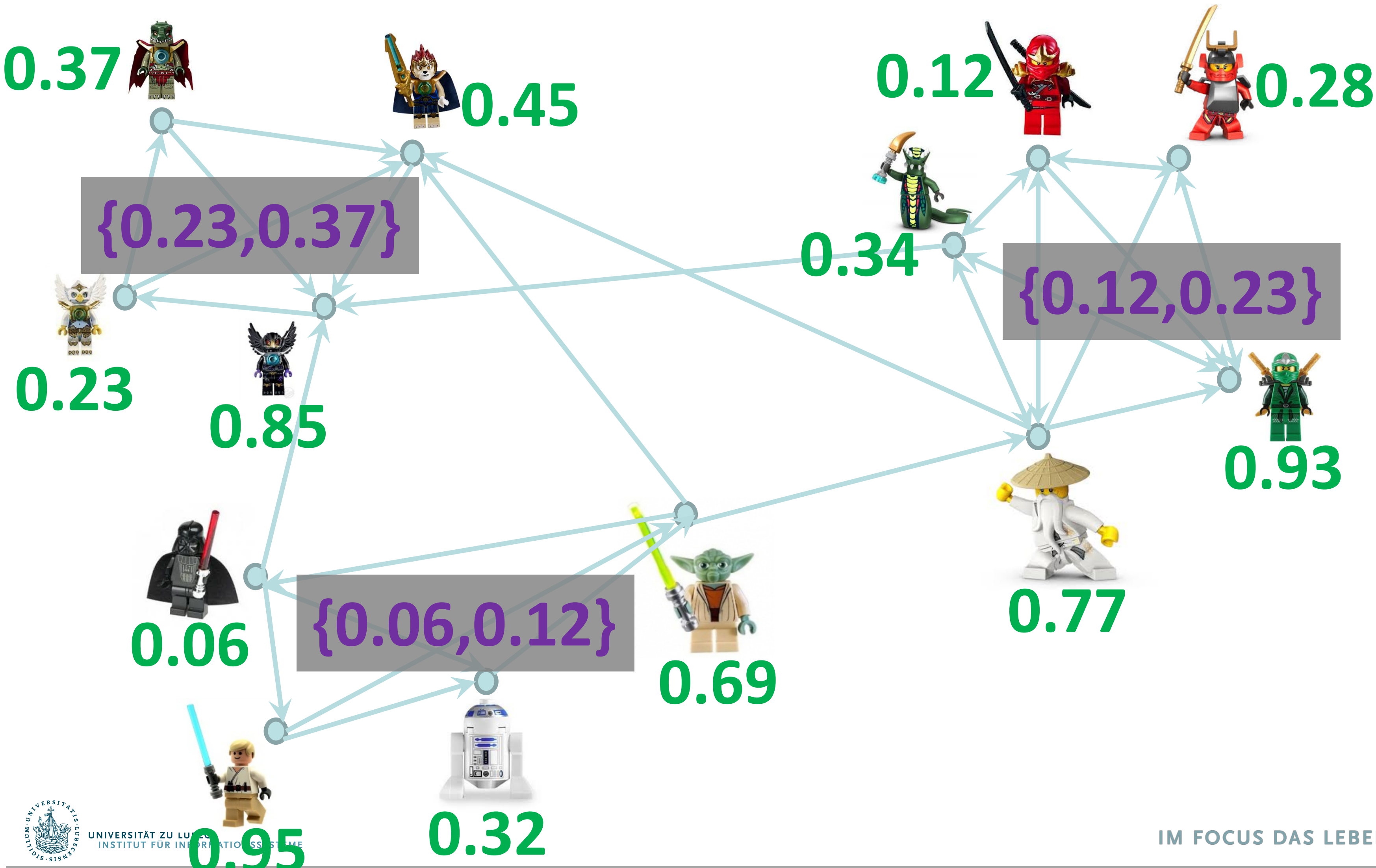
- A group of nodes with the same min-hash value means that there seems to be one node (the one associated with the min-hash initially) that can be reached by all other group nodes
- It may happen, however, that two non-connected nodes have the same min-hash value due to initial hash collisions (false positives w.r.t. reachability are possible)
- Min-hash values allow us to identify groups with important locally central nodes (group hubs)
  - Identify 3 groups (with 0.06, 0.12, and 0.23 nodes as hubs)
- With k=1, one cannot easily see which group can reach which other group

Min-Hash sketches of all Reachability Sets: $k = 2$

For each v: $\mathbf{s}(v) \leftarrow \underset{v \leadsto u}{\mathbf{bottom\text{–}2}}\, h(u)$

bottom-2 ($k = 2$)

Min-Hash sketches of all Reachability Sets: $k = 2$



Determine Groups

- K=2:
  - Identify 3 groups
    - {0.06, 0.12}, {0.12, 0.23}, and {0.23, 0.27}
  - Group 0.06 can reach group 0.12 but not vice versa
    - Overlap 0.12 is min-hash-2 in 0.06
  - Group 0.12 can reach group 0.23 but not vice versa
    - Analogous argument
  - Group 0.23 cannot reach another group
    - For min-hash-2 0.37 there is no overlap

Estimating Jaccard Similarity of Nodes

Assume that k>>2

Min-hash overlap large for two nodes u and v
→ similar influences from other nodes/groups

Nodes u and v can be seen as similar
(note: there are false positives)

Approximate Jaccard by fraction of identical min-hash values

Goal

Computing Min-Hash sketches of all reachability sets efficiently

Sketch size for a node: $O(k)$

Total computation $\approx O(km)$

Algorithms/methods:

- Graphs searches (say BFS)
- Dynamic programming / Distributed

$k = 1$ BFS method

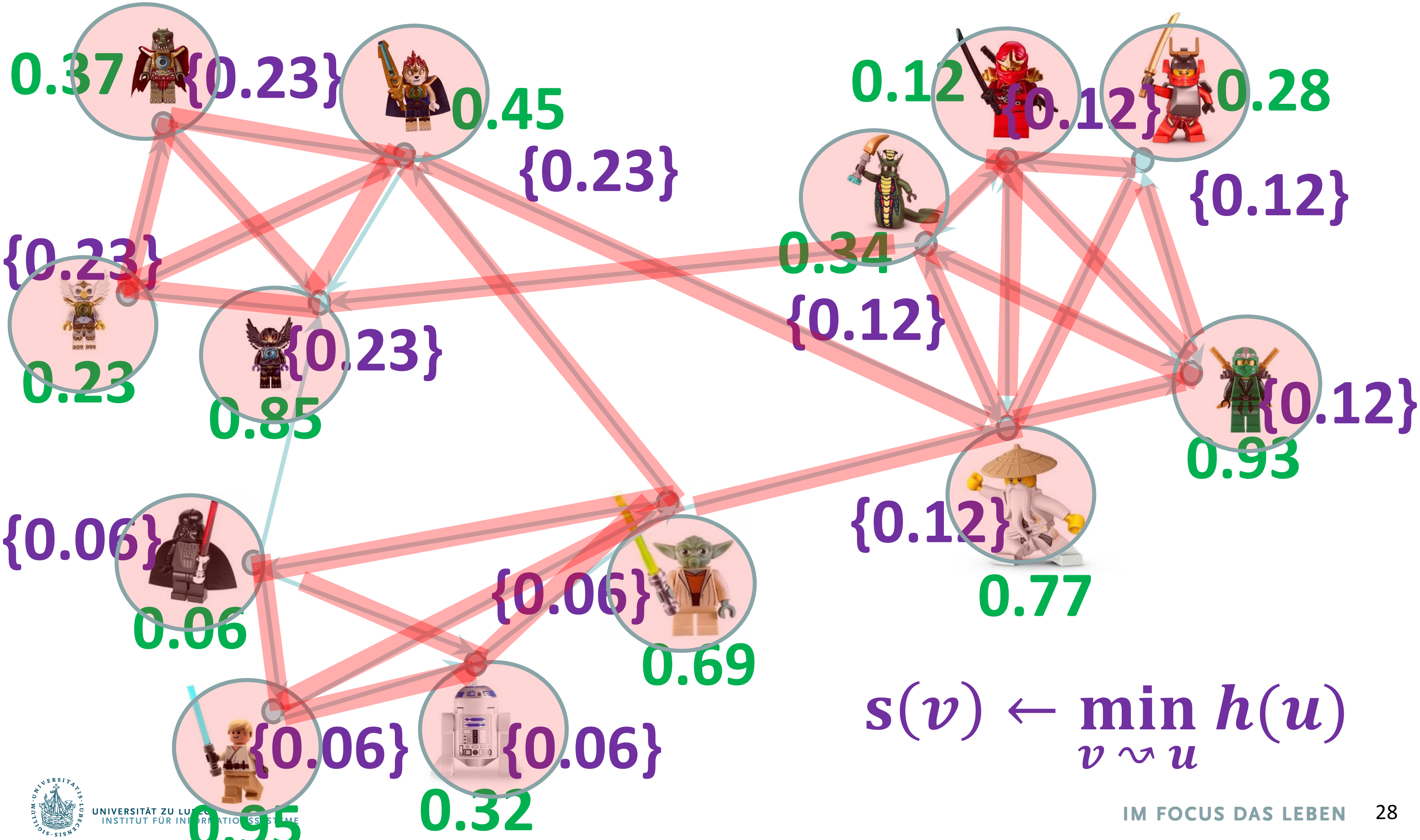
$$\mathbf{s}(\boldsymbol{v}) \leftarrow \min_{\boldsymbol{v} \rightsquigarrow \boldsymbol{u}} \boldsymbol{h}(\boldsymbol{u})$$

Iterate over nodes u by increasing $h(u)$:

Visit nodes v through a reverse search from u:

- **IF** $s(v) = \emptyset$,
  - $s(v) \leftarrow h(u)$
  - Continue search on $\text{inNeighbors}(v)$
- **ELSE** truncate search at v

Min-Hash sketches: $k = 1$, BFS



$$s(v) \leftarrow \min_{v \rightsquigarrow u} h(u)$$

Min-Hash-BFS Analysis

- Each arc is used exactly once: $O(m)$
- Each graph search depends on all previous ones: seems like we need to perform n searches sequentially
- How can we reduce dependencies ?

Parallel BFS-based Min-Hash

Idea ($k = 1$):

- Create a super-node of the $n/2$ lowest hash nodes.
- Perform a (reverse) search from super-node and mark all nodes that are accessed.
- Concurrently perform searches:
  - From the lowest-hash $n/2$ nodes (sequentially)
  - From the highest-hash $n/2$ (sequentially). Prune searches **also** at marked nodes

Parallel BFS-based Min-Hash

Correctness:

- **For the lower $n/2$ hash values**: computation is the same.
- **For the higher $n/2$:**

  We do not know the minimum reachable hash from higher-hash nodes, but we do know it is one of the lower $n/2$ hash values. This is all we need to know for correct pruning.

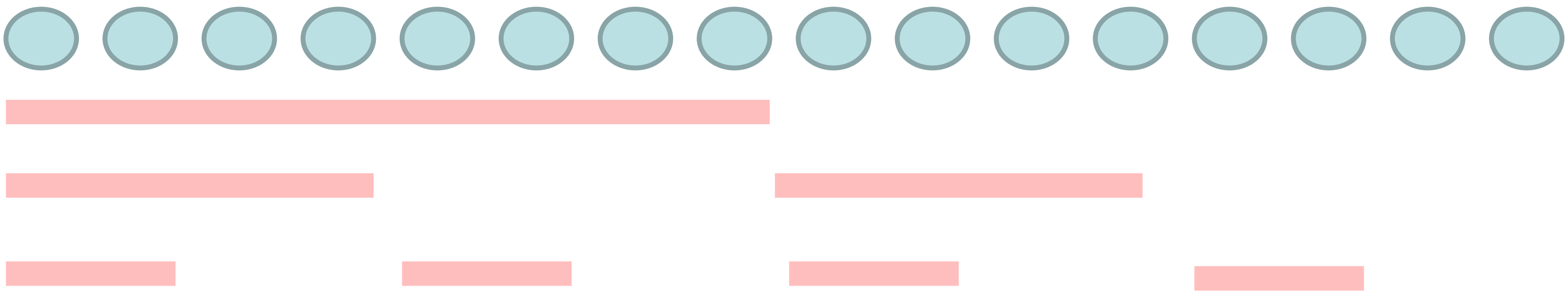
Parallel BFS-based Min-Hash: Analysis

- This only gives us $n/2$ instead of n sequential searches.

How can we obtain more parallelism ?

- We **recursively** apply this to each of the lower/higher sets:

Parallel BFS-based Min-Hash



Nodes ordered by $h(u)$

Super-nodes created in recursion

- The depth of dependencies is at most $\log_2 n$
- The total number of edge traversals can increase by a factor of $\log_2 n$

Computing Min-Hash Sketches of all Reachability Sets

bottom-k, BFS method

Next: Computing sketches using the BFS method for k>1

$$\mathbf{s}(v) \leftarrow \underset{v \leadsto u}{\textbf{bottom-}k} \ h(u)$$

Computing Min-Hash Sketches of all Reachability Sets

bottom-k, BFS method

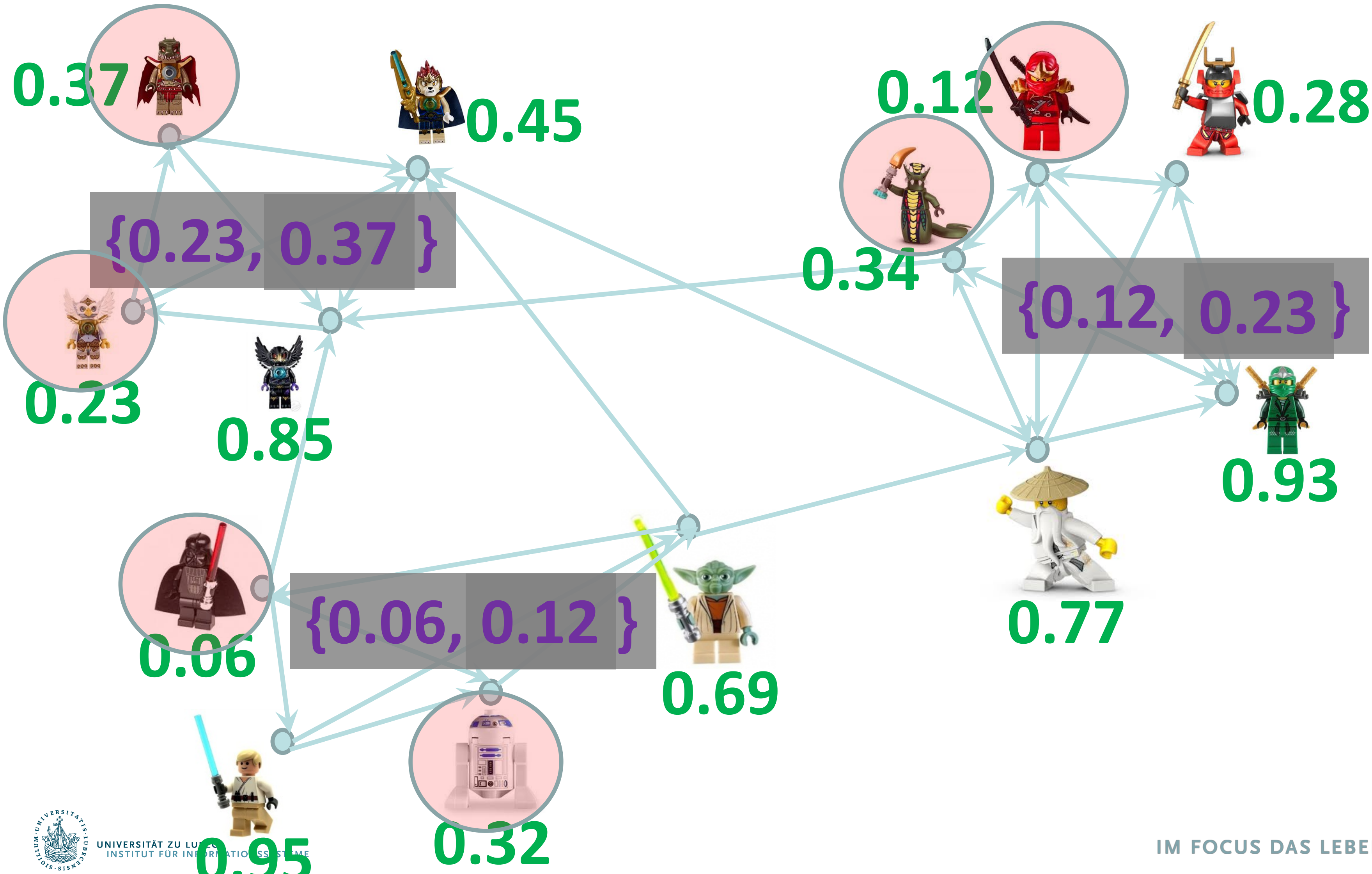
$$\mathbf{s}(\boldsymbol{v}) \leftarrow \underset{\boldsymbol{v} \rightsquigarrow \boldsymbol{u}}{\textbf{bottom-}\boldsymbol{k}}\ \boldsymbol{h}(\boldsymbol{u})$$

Iterate over nodes u by increasing $h(u)$:

Visit nodes v through a reverse search from u:

- **IF** $|\mathrm{s}(v)| < k$,
  - $s(v) \leftarrow s(v) \cup \{h(u)\}$
  - Continue search on $\mathrm{inNeighbors}(v)$
- **ELSE** truncate search at v

Min-Hash sketches of all Reachability Sets: bottom-2



Computing Min-Hash Sketches of all Reachability Sets

$k = 1$ Distributed (DP)

Next: back to $k = 1$.
We present **another method** to compute the sketches. The algorithm has fewer dependencies. It is specified for each node. It is suitable for computation that is:

- Distributed, Asynchronous
- Dynamic Programming (DP)
- Multiple passes on the set of arcs

Computing Min-Hash Sketches of all Reachability Sets:

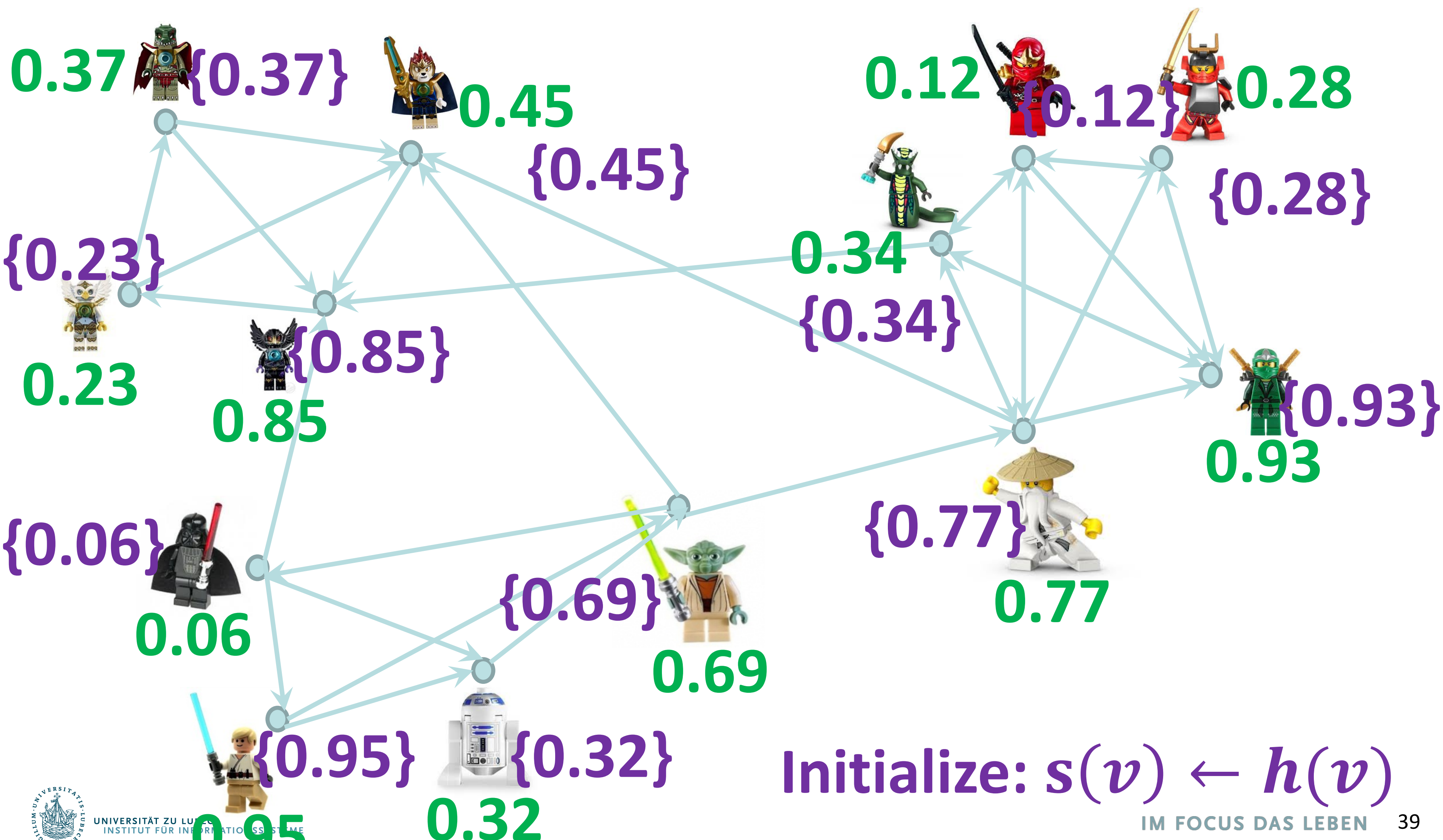
$k = 1$ Distributed (DP)

$$\mathbf{s}(\boldsymbol{v}) \leftarrow \min_{\boldsymbol{v} \rightsquigarrow \boldsymbol{u}} \boldsymbol{h}(\boldsymbol{u})$$

Initialize $\mathbf{s}(\boldsymbol{v}) \leftarrow \boldsymbol{h}(\boldsymbol{v})$

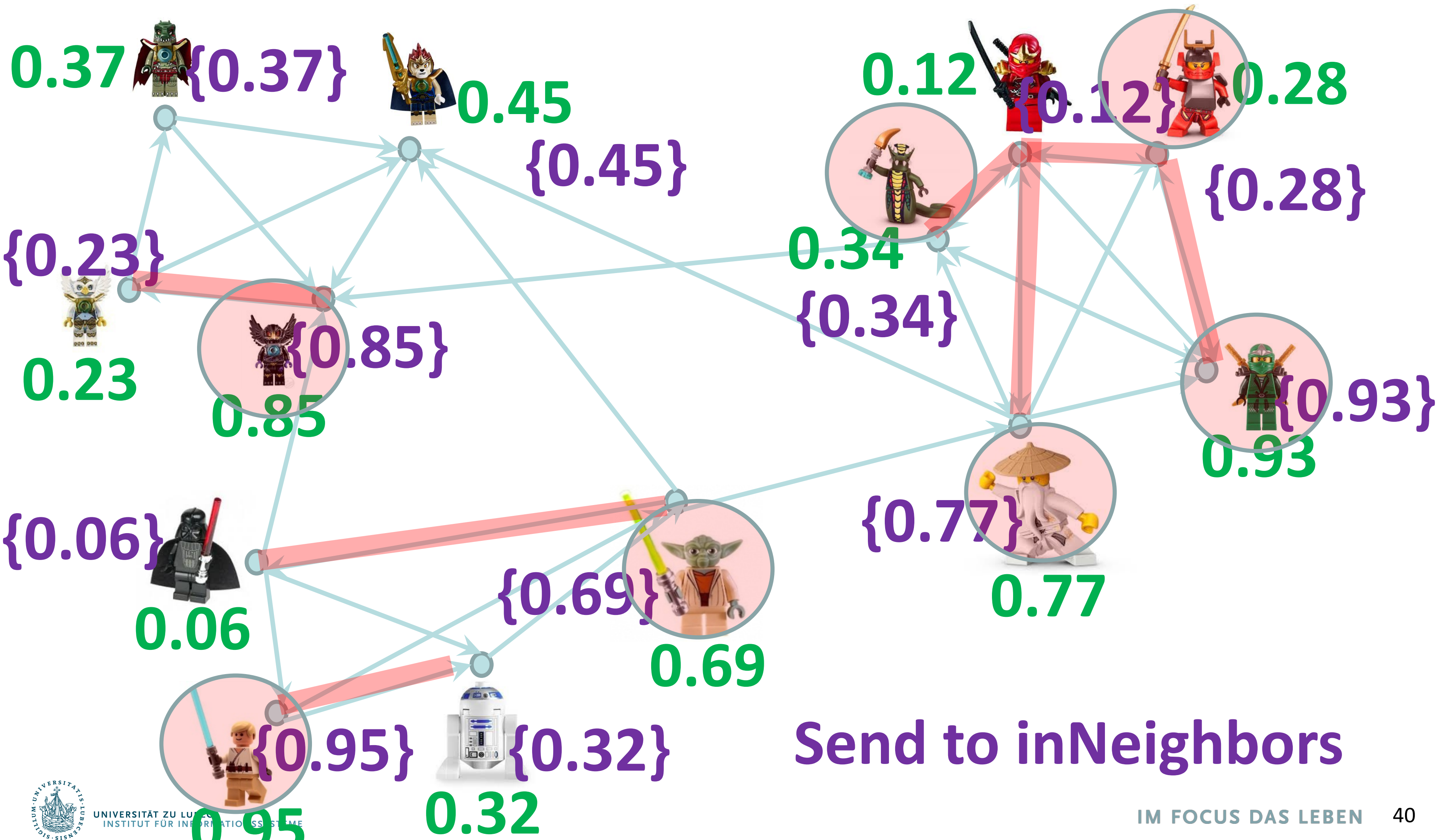
- **IF** $s(v)$ is initialized/updated, send $s(v)$ to $\text{inNeighbors}(v)$
- **IF** value x is received from neighbor:
  - $s(v) \leftarrow \min\{s(v), x\}$

DP computation of Min-Hash sketches $k = 1$



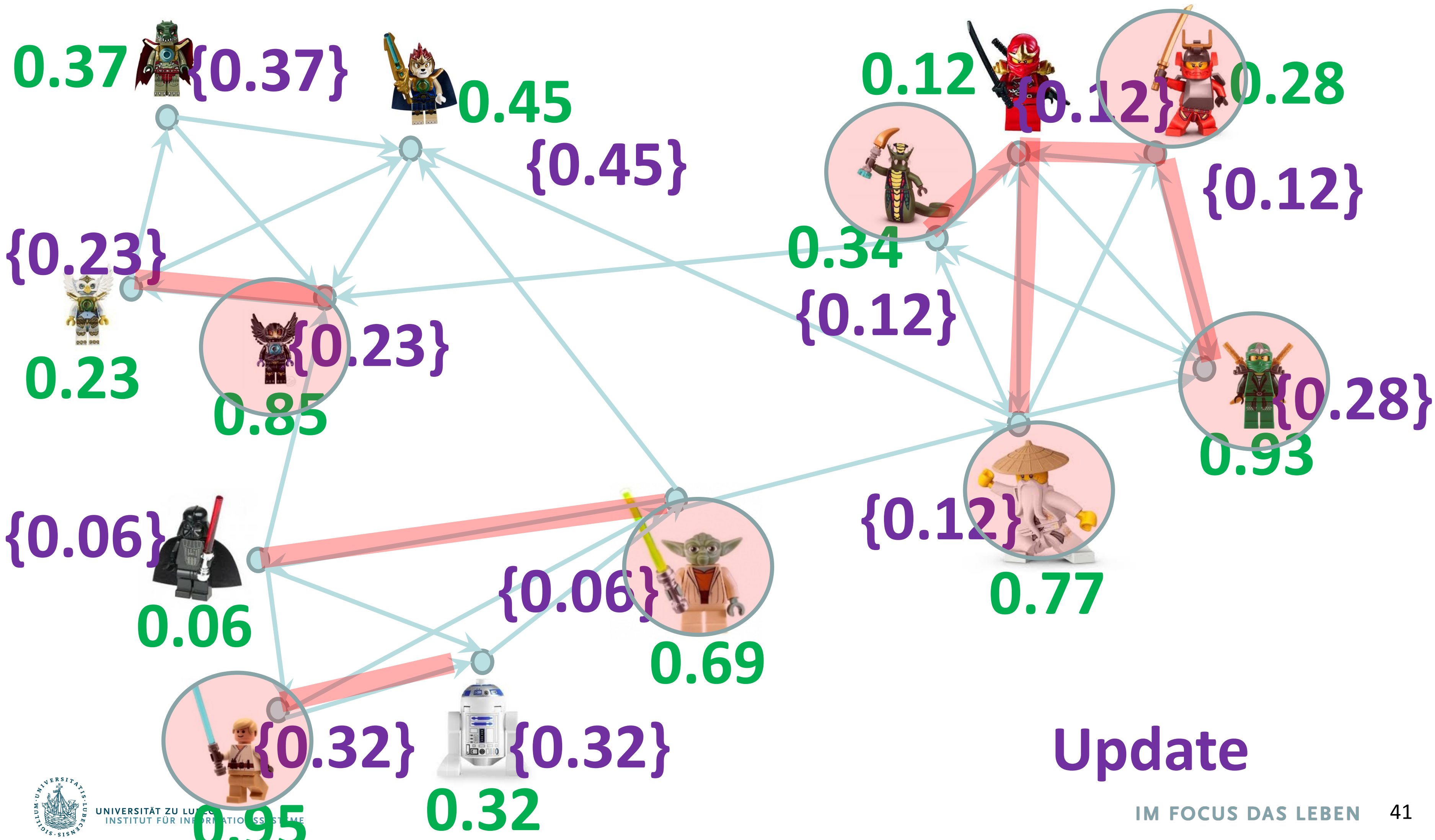
Initialize: $s(v) \leftarrow h(v)$

DP computation of Min-Hash sketches $k = 1$



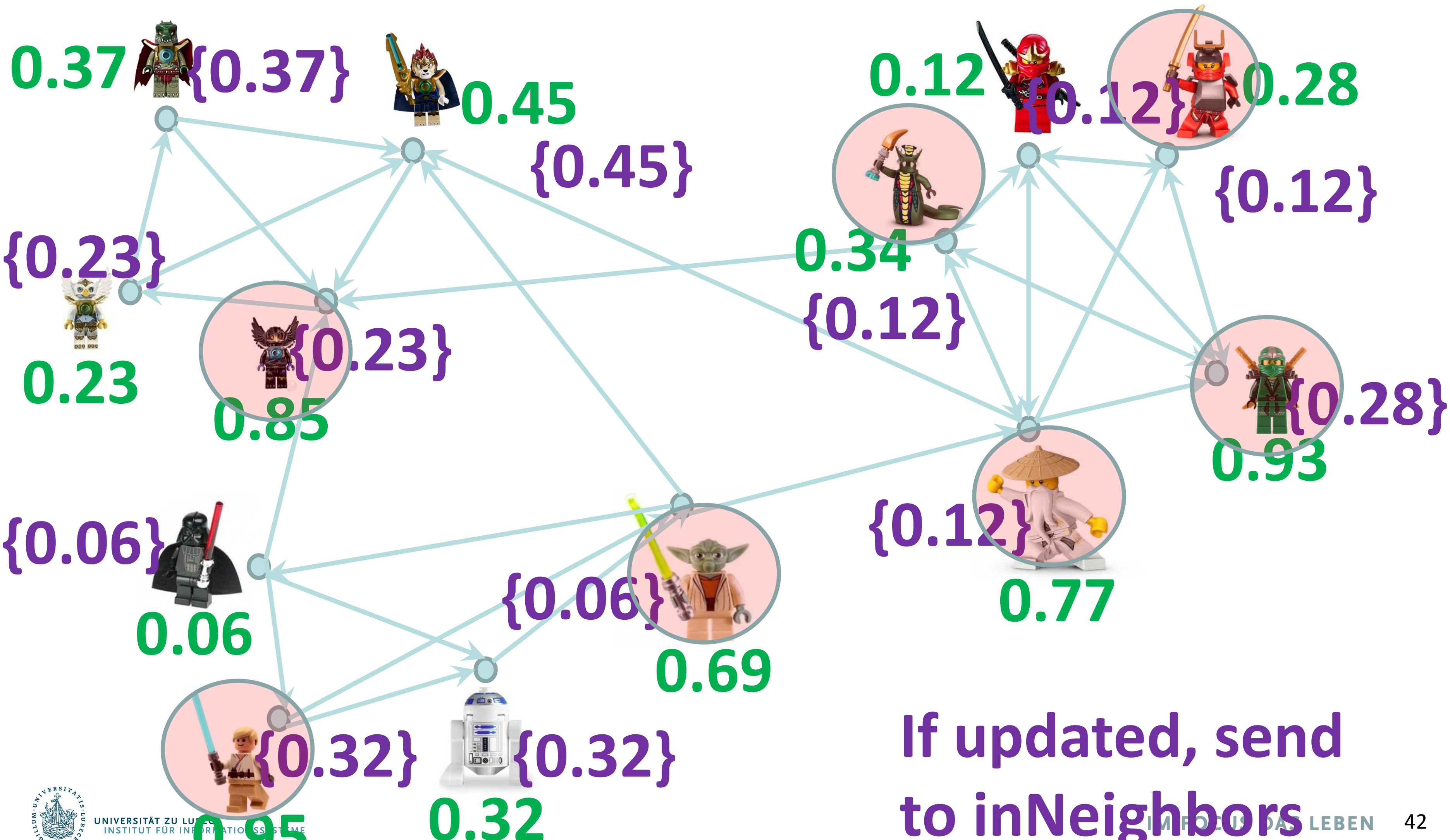
Send to inNeighbors

DP computation of Min-Hash sketches $k = 1$



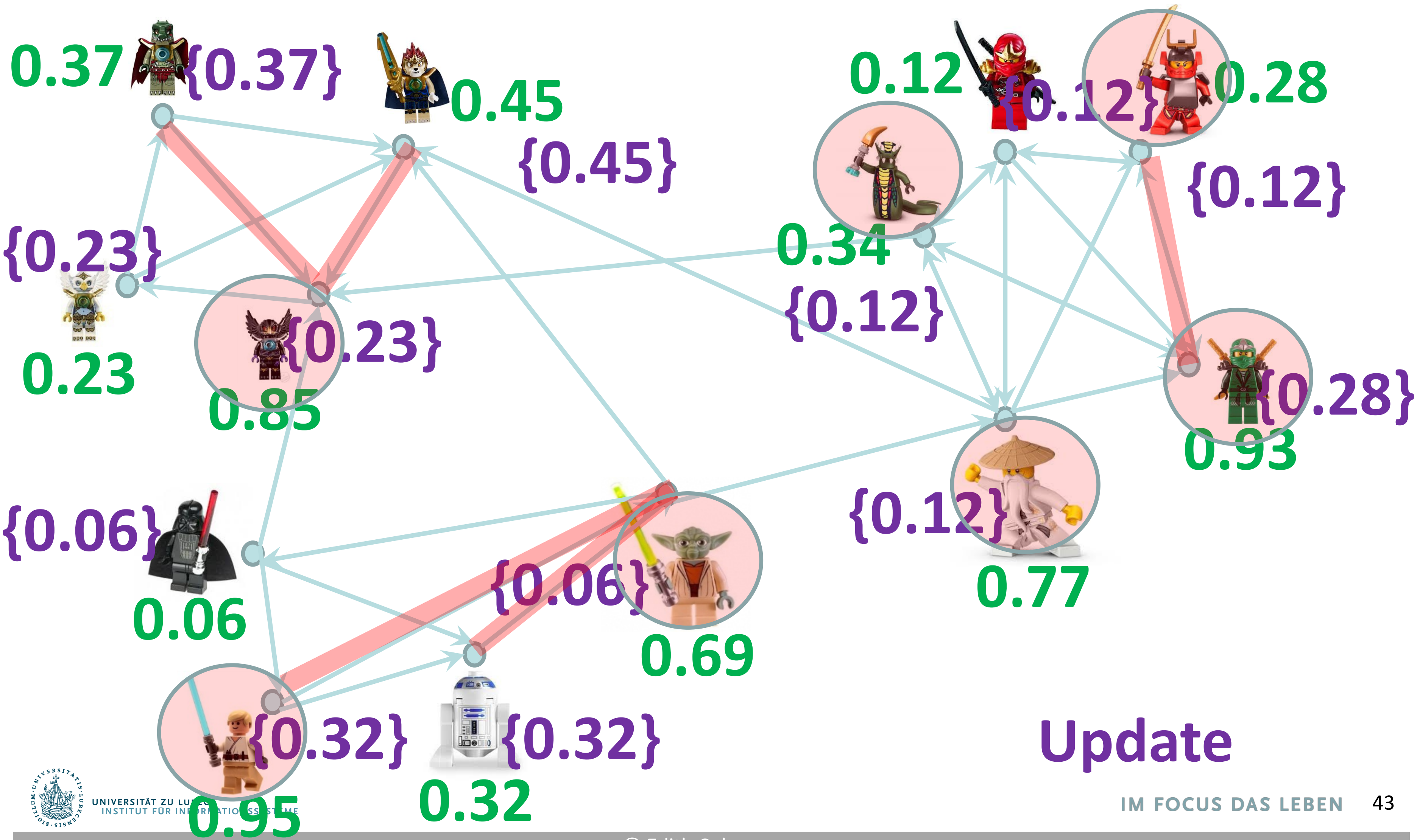
Update

DP computation of Min-Hash sketches $k = 1$

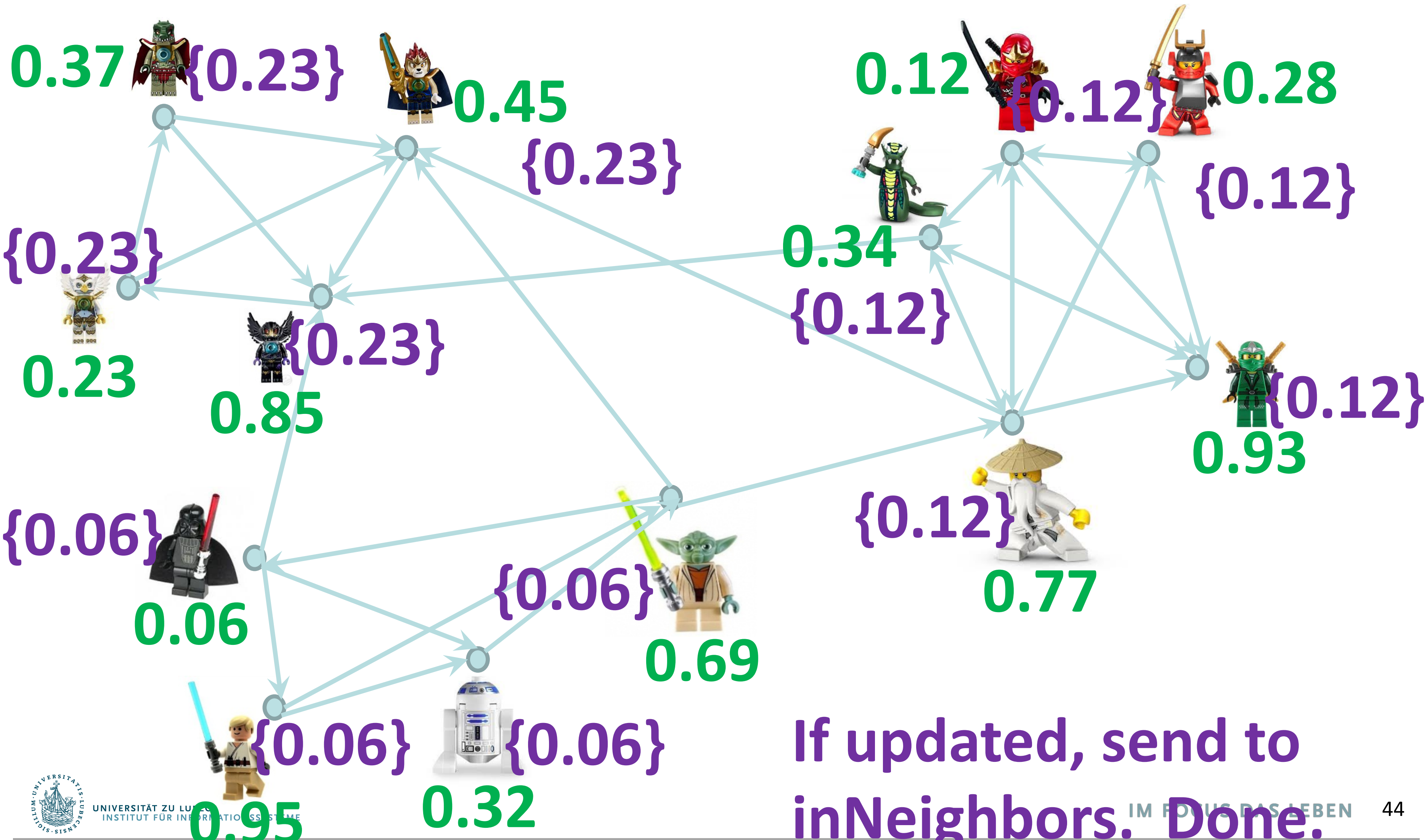


If updated, send to inNeighbors

DP computation of Min-Hash sketches $k = 1$



DP computation of Min-Hash sketches $k = 1$



If updated, send to inNeighbors. Done.

Analysis of DP: Edge traversals

Lemma: Each arc is used in expectation $< \ln \boldsymbol{n}$ times.

Proof: We bound the expected number of updates of $\boldsymbol{s}(\boldsymbol{v})$

- Consider nodes $v = u_1, u_2, \ldots$ in order that $h(u_i)$ is propagated to (can reach) v.
- The probability that $\mathrm{h}(u_i)$ updates $\mathrm{s}(v)$:

$$\mathbf{Pr}[\boldsymbol{h}(\boldsymbol{u}_i) < \min_{j<i} \boldsymbol{h}(\boldsymbol{u}_j)] = \frac{1}{i}$$

- Summing over nodes (linearity of expectation):

$\sum_{i=1}^{n} \frac{1}{i} = \boldsymbol{H}_n < \ln \boldsymbol{n}$

Harmonische Reihe

Analysis of DP: dependencies

The longest chain of dependencies is at most the longest shortest path (the diameter of the graph)

All-Distances Sketches (ADS)

Often we care about distance, not only reachability:

- Nodes that are closer to a particular, in distance or in Dijkstra (Nearest-Neighbor) rank, are more meaningful for the node
- We want a sketch that supports distance-based queries (node hops)
- ADS-Sketch: Inclusion probability of the min-hash of a node u decreases with its distance from v (more precisely, inversely proportional to the number of nodes closer to v than u)
- Estimating similarity between neighborhoods of two nodes, distances, closeness similarities, etc.

Edith Cohen. All-distances sketches, revisited: HIP estimators for massive graphs analysis. In Proc. Symposium on Principles of database systems (PODS '14). ACM, New York, NY, USA, 88-99, **2014**