Übung zu Peer-to-Peer und Cloud Computing

Übungstermin 06: Besprechung des Übungsblattes 05

Dominik Rauh

12. Dezember 2018

Universität Augsburg
Institut für Informatik
Lehrstuhl für Organic Computing

Gegebenes CAN

ORGANIC COMPUTING

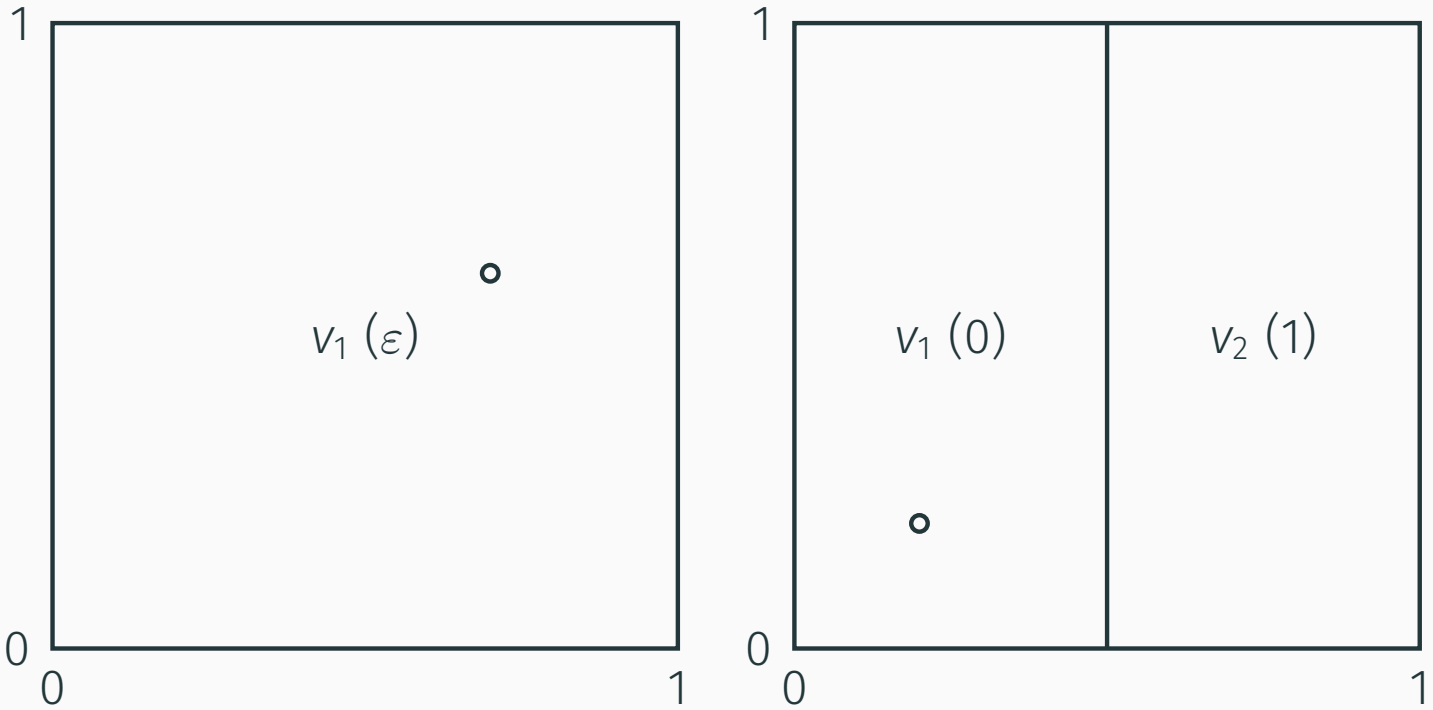
- leeres CAN
- Zonenteilung
  - in der Mitte der längeren Seite
  - sind die Seiten gleich lang, in der ersten Dimension (also *vertikal*)
- Teilzone neuer Knoten
  - bei vertikaler Teilung: rechts
  - bei horizontaler Teilung: oben

In dieses CAN ordnen sich nun nacheinander neun Knoten ein. Zeichnen Sie die zweidimensionale Struktur des Netzwerks nach jedem neu hinzugefügten Knoten!

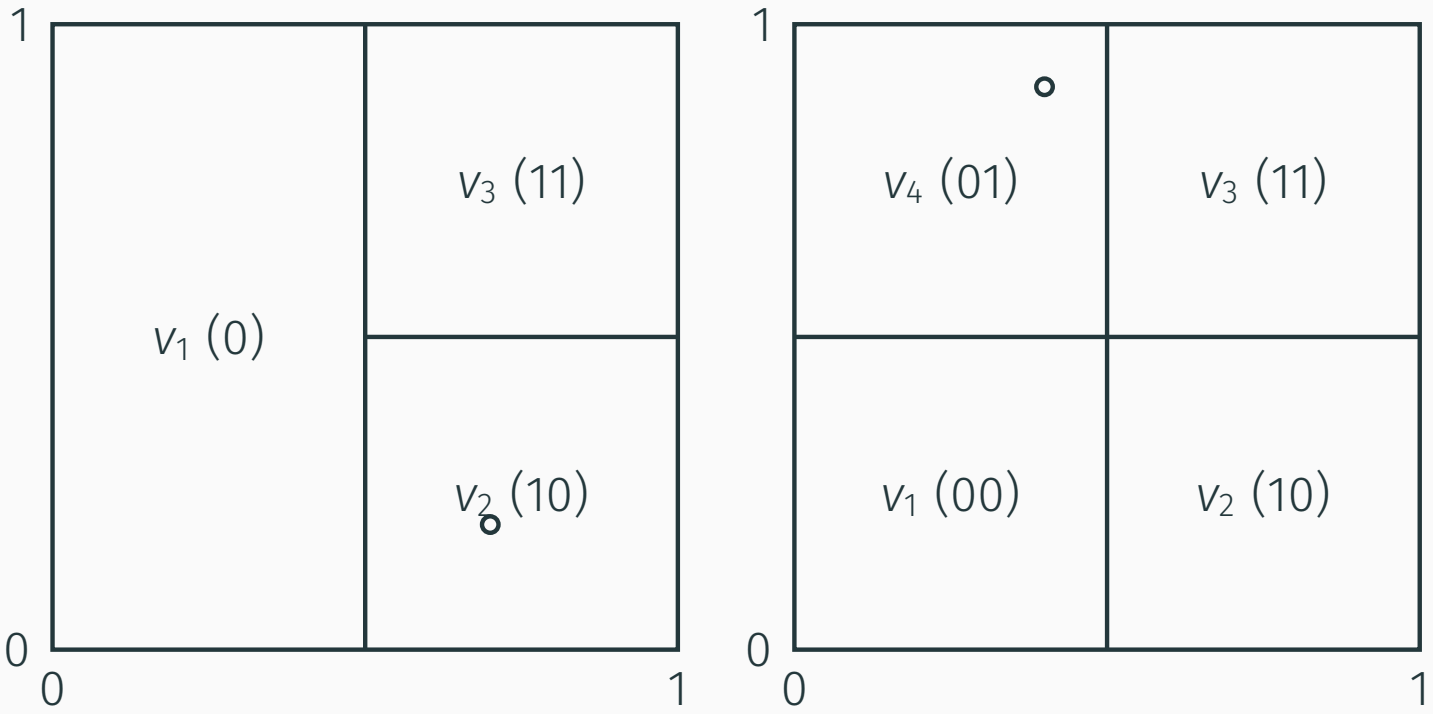
Sich einordnende Knoten

- $v_1 = (0{,}70; 0{,}60)$
- $v_2 = (0{,}20; 0{,}20)$
- $v_3 = (0{,}70; 0{,}20)$
- $v_4 = (0{,}40; 0{,}90)$
- $v_5 = (0{,}90; 0{,}90)$
- $v_6 = (0{,}90; 0{,}40)$
- $v_7 = (0{,}40; 0{,}40)$
- $v_8 = (0{,}40; 0{,}60)$
- $v_9 = (0{,}20; 0{,}60)$

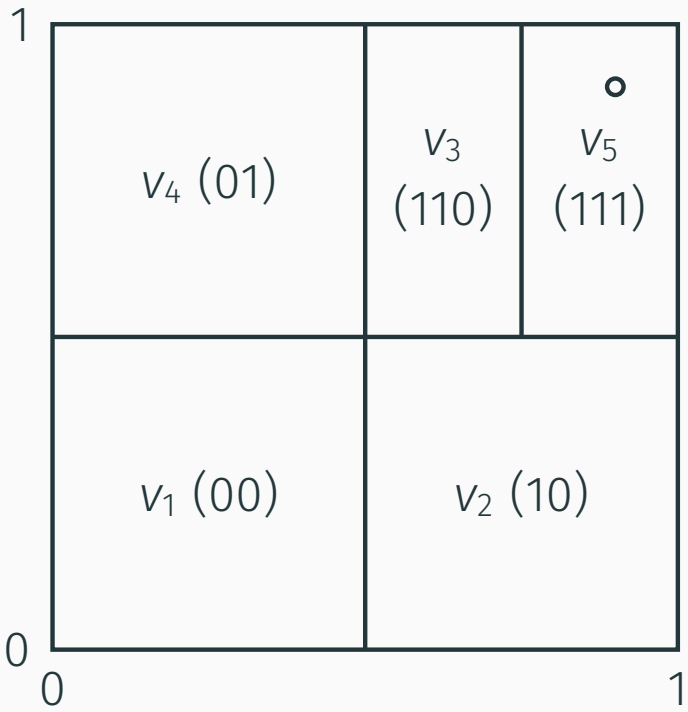
Struktur des CANs i

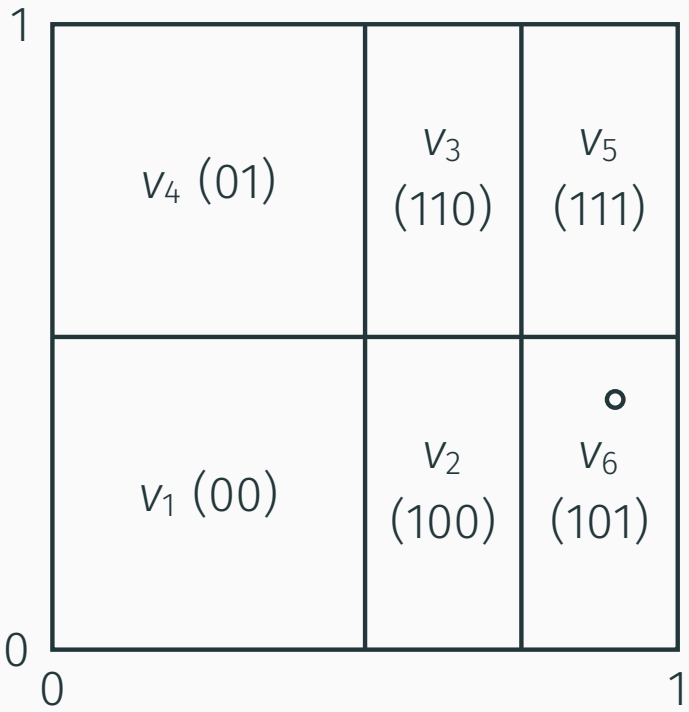


Struktur des CANs ii

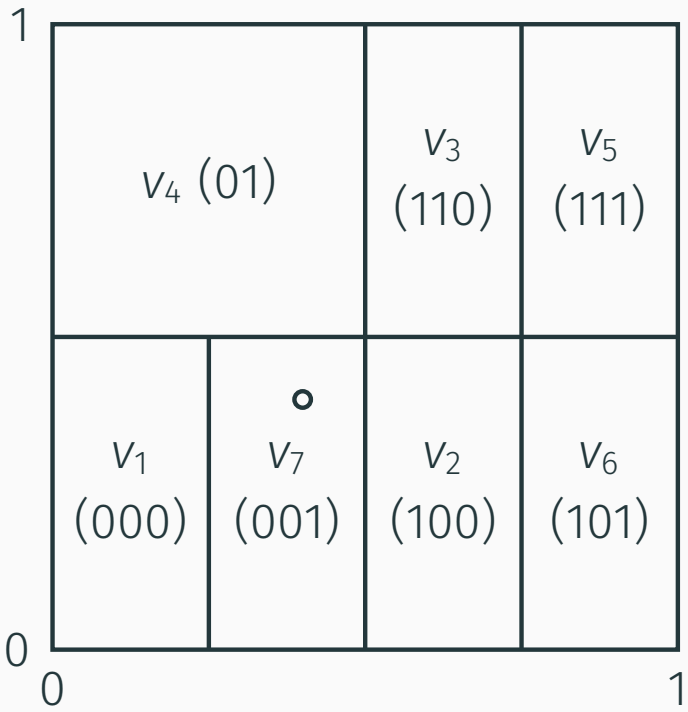


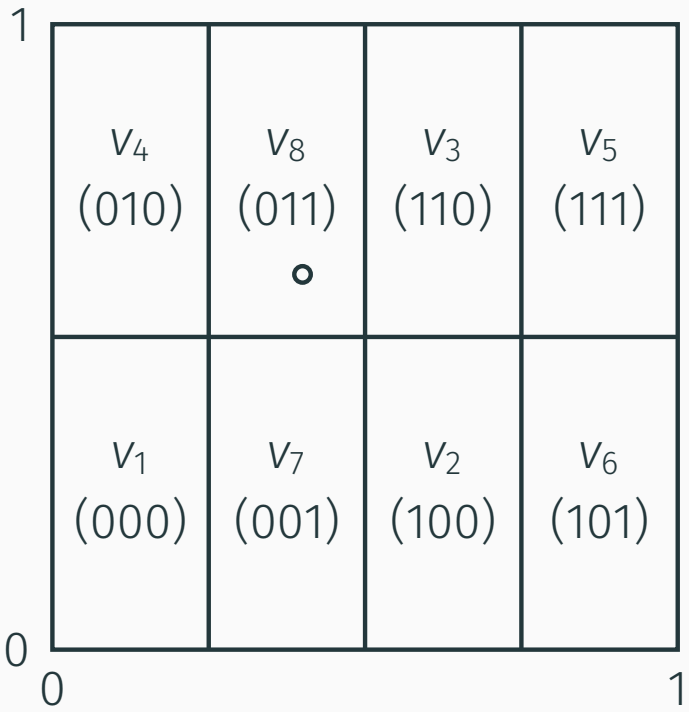
Struktur des CANs iii



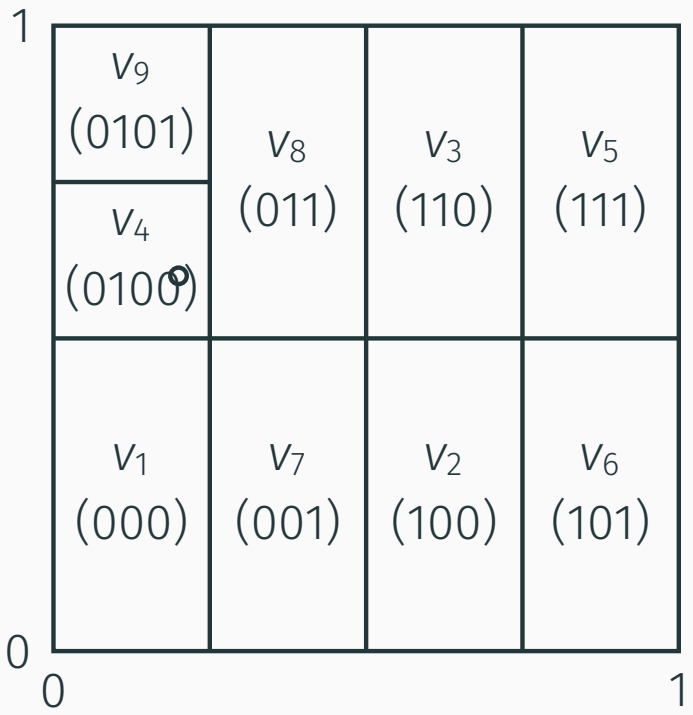


Struktur des CANs iv



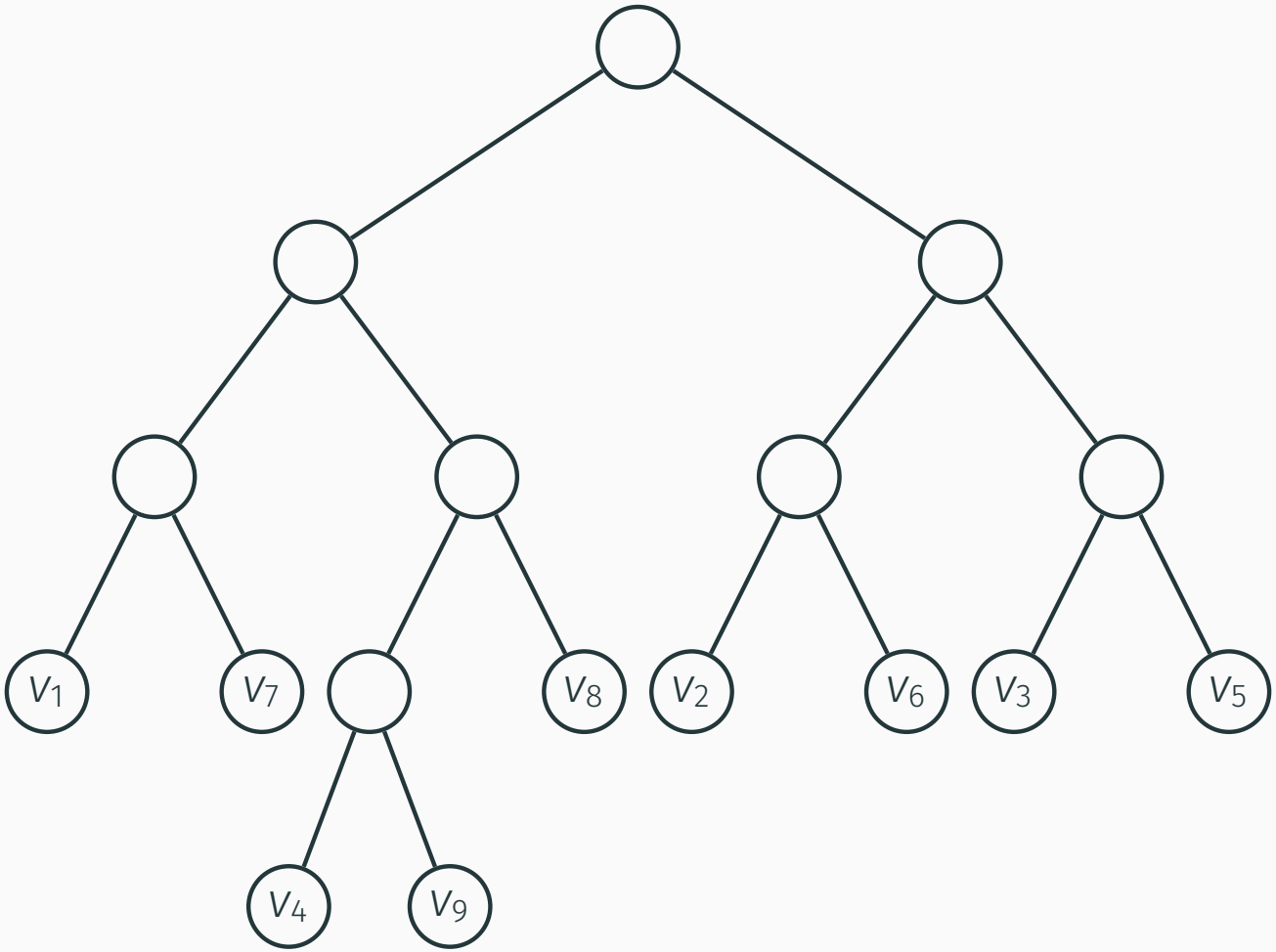


Struktur des CANs v



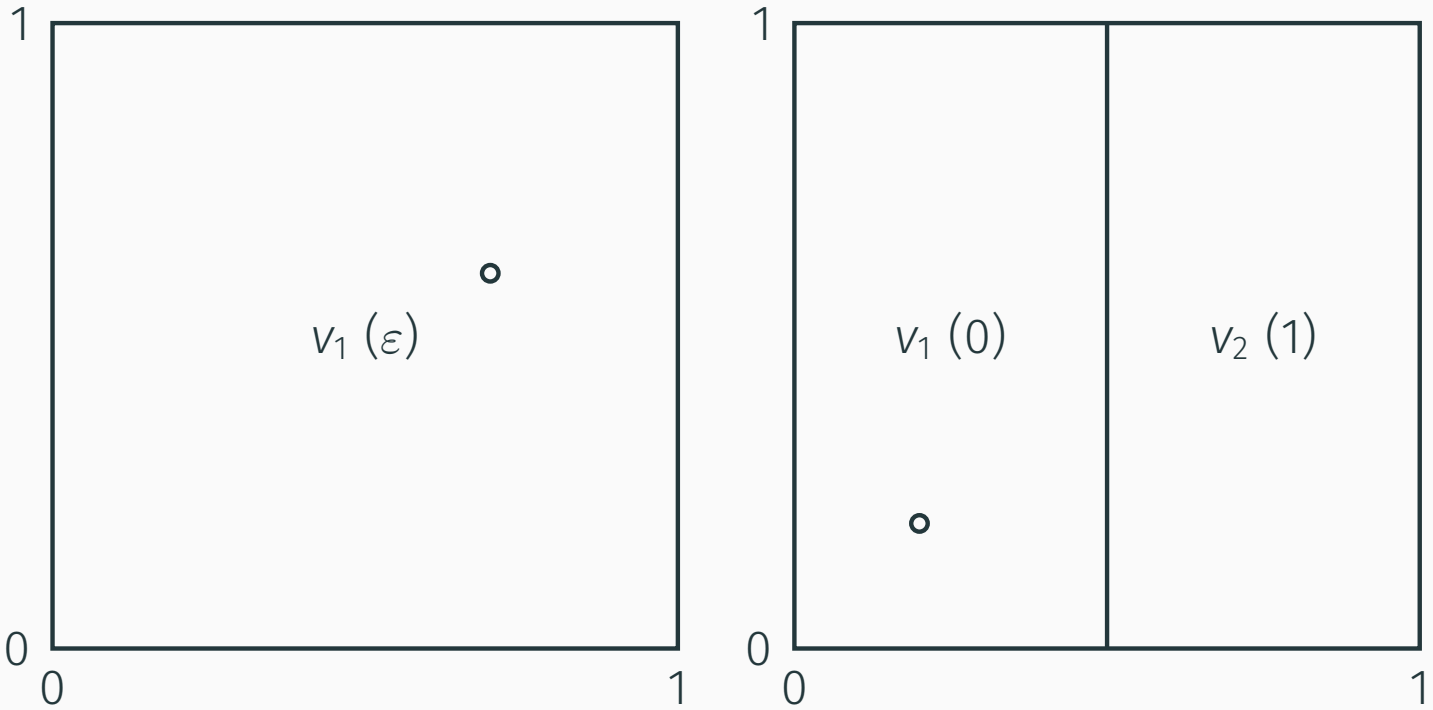
Zeichnen Sie den Partitionsbaum des endgültigen Netzwerks.

Partitionsbaum des CANs

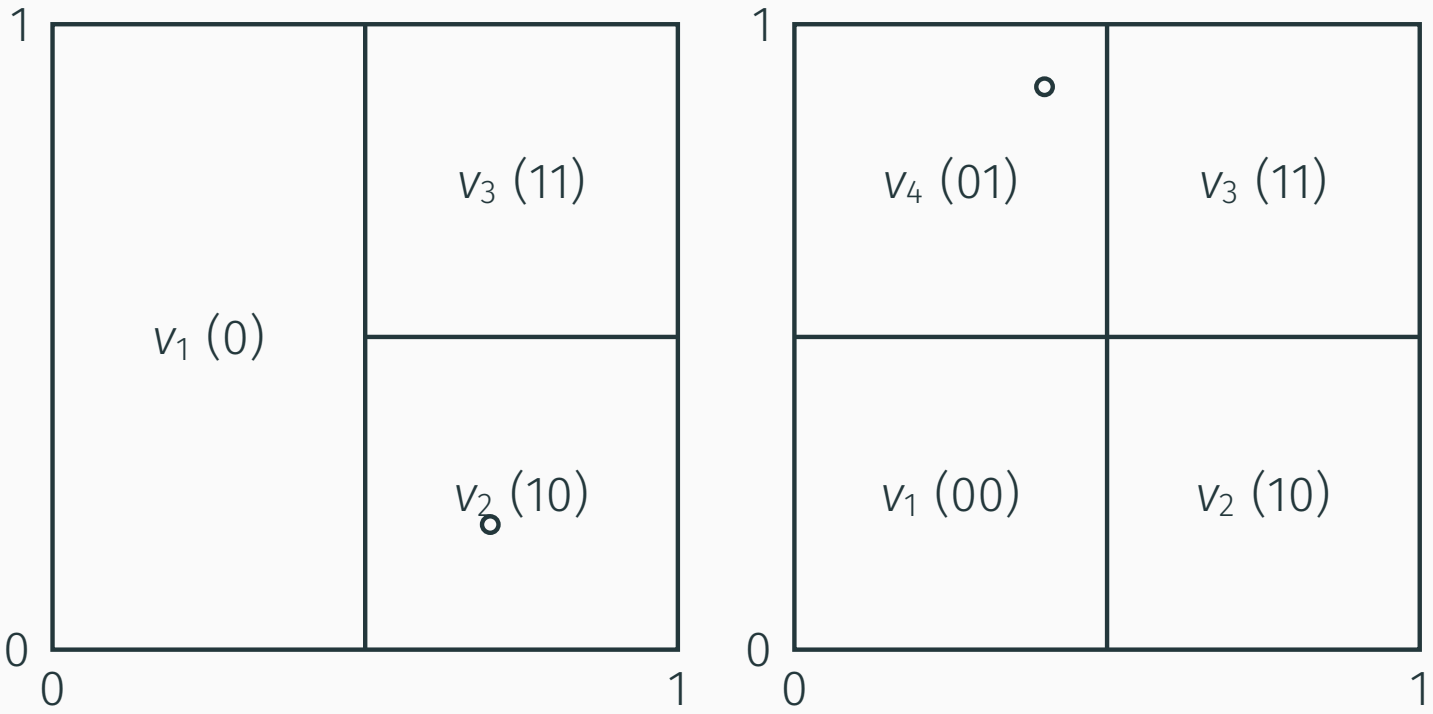


Zeichnen Sie die *virtual IDs* (VIDs) in alle Zwischenschritte ein.

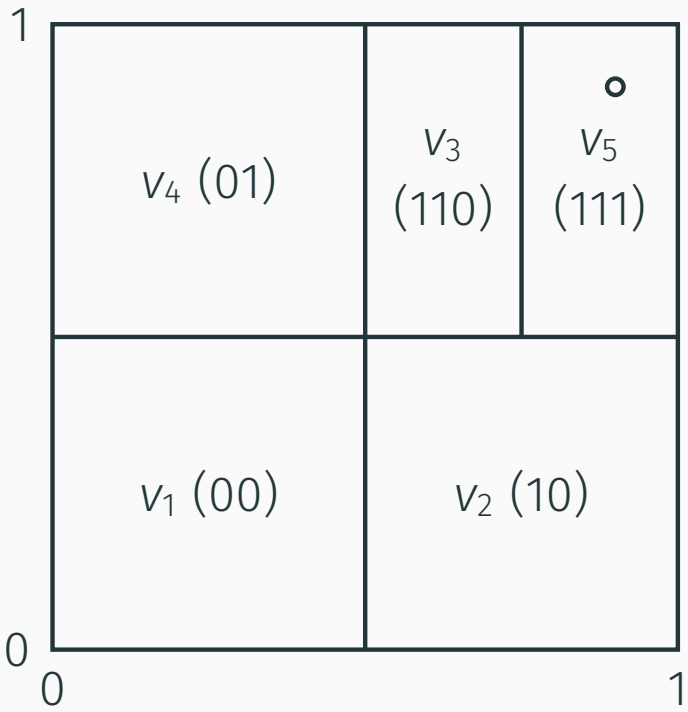
VIDs des CANs i

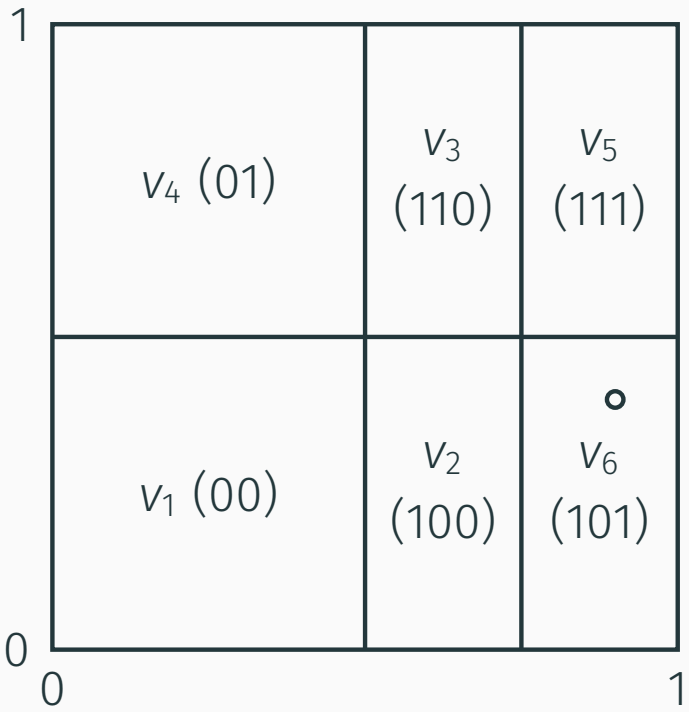


VIDs des CANs ii

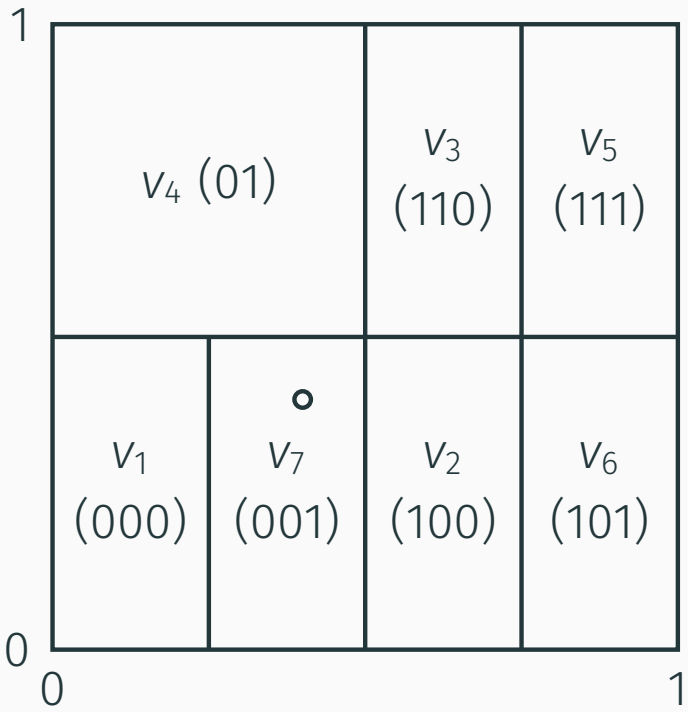


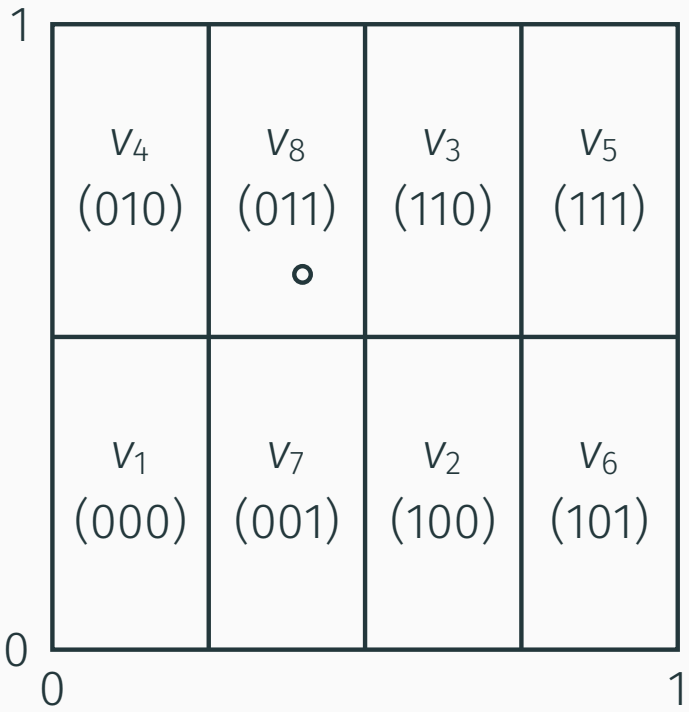
VIDs des CANs iii



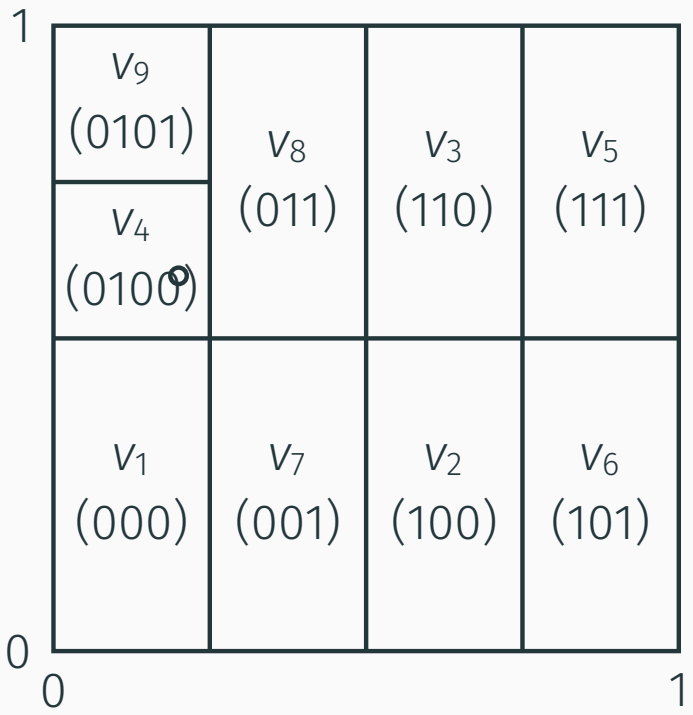


VIDs des CANs iv





VIDs des CANs v



v_8 meldet seinen Austritt. Auf welche Weise *sollte* sich das Netzwerk verändern?

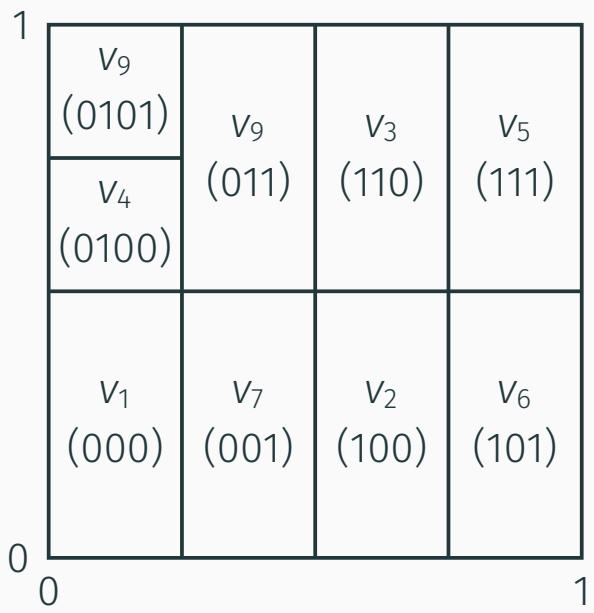
Takeover-Knoten von v_8

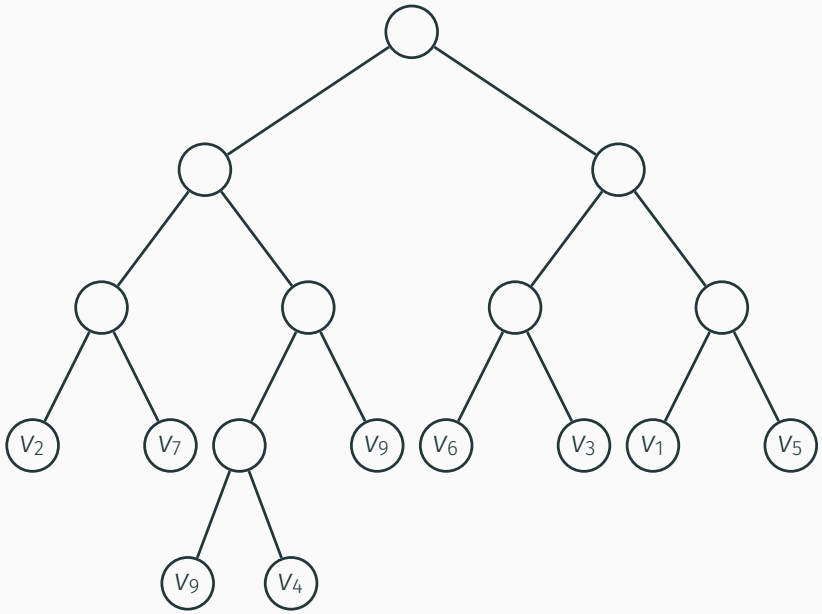
ORGANIC COMPUTING

- Geschwisterteil v von v_8 ist *kein* Blatt
- somit Tiefensuche im Teilbaum $t_v \setminus \{v_8\}$ (bis Blatt gefunden)
- v_9 wird als Blatt gefunden (oder v_4)
  $\Rightarrow$ v_9 wird Takeover-Knoten
- Zusammenführen der Zonen von v_8 und v_9 nicht möglich
  $\Rightarrow$ zwei neue Netzwerkstrukturen möglich

Zonenmanagement: Möglichkeit 1

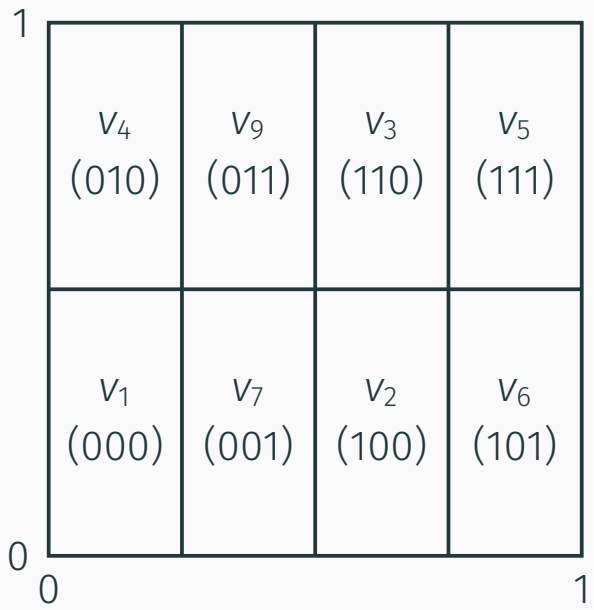
v_9 managt Zone von v_8 bis neuer Knoten in v_9 beitritt und übergibt diese an ihn

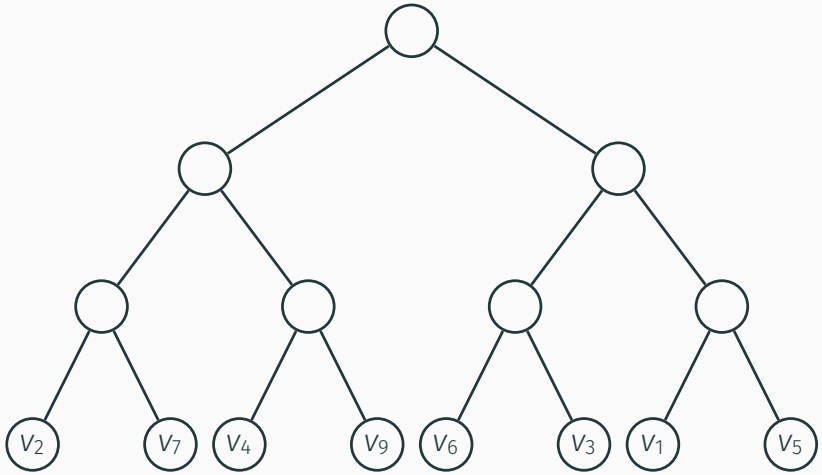




Zonenmanagement: Möglichkeit 2

die Zone von v_9's Geschwisterteil v_4 wird mit der von v_9 zusammengelegt und nun von v_4 verwaltet; v_9 übernimmt Zone von v_8





In Echtwelt-Netzwerken kennen die Knoten den Partitionsbaum nicht, sondern nur die VIDs ihrer Nachbarn. Beschreiben Sie den Ablauf des Austritts von v_8, wenn für die Recovery-Nachrichten *Greedy Forwarding* benutzt wird!

Zwei mögliche Antworten!

Fall: v_8 verlässt das Netzwerk *gracefully*

ORGANIC COMPUTING

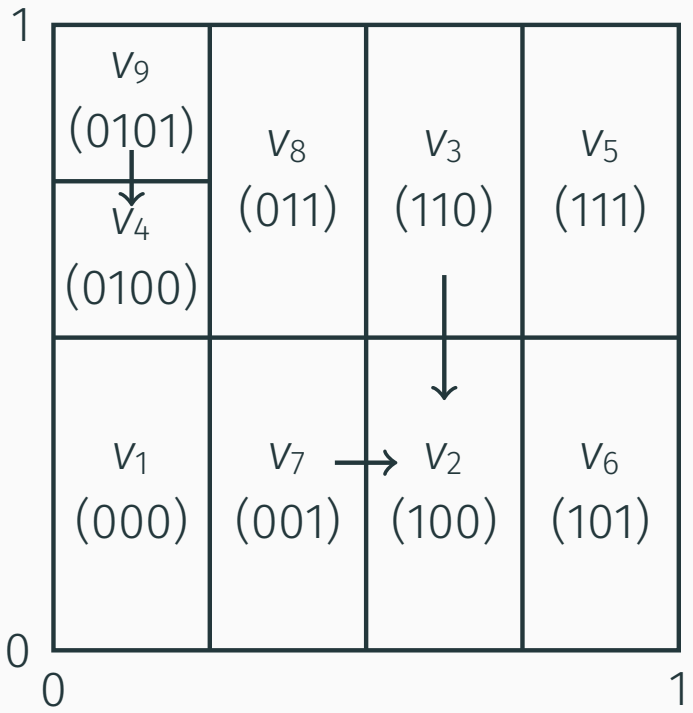
- kann v_9 (oder v_4) vorher noch direkt ansprechen
- v_9 (oder v_4) übernimmt wie in a) beschrieben die Zone von v_8
- v_8 kann auch noch seine anderen Nachbarn über seine Wahl des Takeover-Knoten benachrichtigen

Fall: v_8 versagt (Nachbarauswahl: numerisch)

Nachbarn (v_3, v_4, v_7, v_9) erkennen Versagen

$\Rightarrow$ Recovery-Nachricht an Nachbarn, dessen VID der von v_8 am nächsten ist

$\Rightarrow$ als Takeover-Knoten werden identifiziert: v_2 und v_4



Alternative Nachbarauswahl

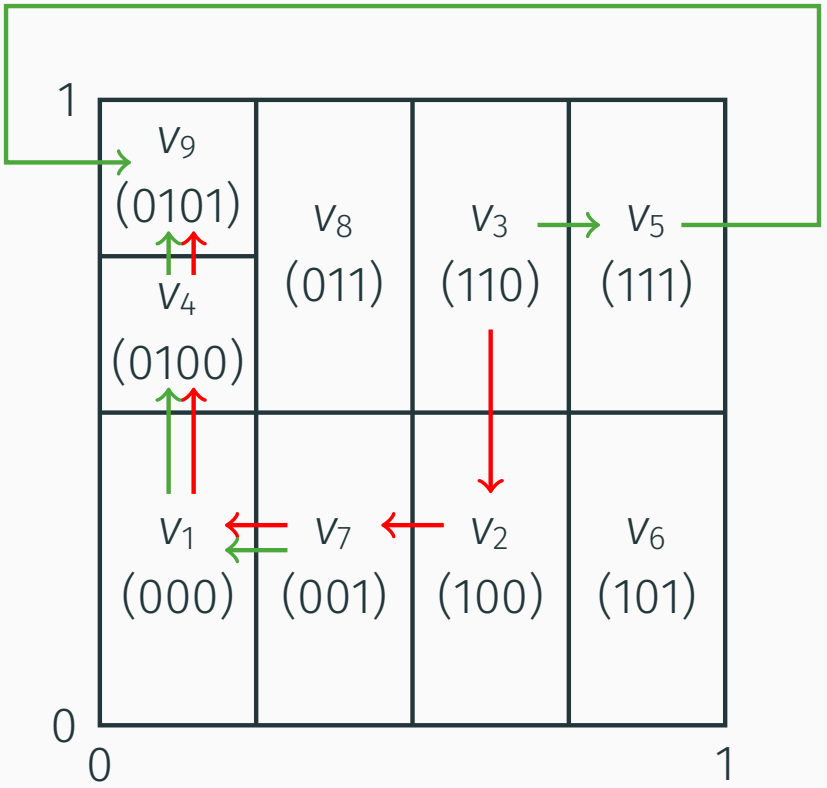
ORGANIC COMPUTING

- längste Präfixübereinstimmung
- Mehrere gleichgeeignete? Zufällige Auswahl!
- Keiner geeigneter aber jemand genauso geeignet wie aktueller Knoten? Zufällige Auswahl eines Nachbarn!
- Keiner geeigneter oder genauso geeignet wie aktueller Knoten? Aktueller Knoten ist Takeover-Knoten!
- Recovery-Nachricht nie zurückschicken

Fall: v_8 versagt (Nachbarauswahl: längstes Präfix)

ORGANIC COMPUTING

Nachbarn (v_3, v_4, v_7, v_9) erkennen Versagen



$\Rightarrow$ als Takeover-Knoten wird identifiziert (unabhängig von Zufall): v_9

Welches Problem tritt dabei auf?

Problem mit VID-basiertem *Greedy Forwarding*

ORGANIC COMPUTING

- bei numerischer Nachbarauswahl: sowohl v_2 und v_4 werden als Takeover-Knoten ausgewählt
  $\Rightarrow$ Inkonsistenz!
- außerdem: Ausfall mehrerer Knoten
  $\Rightarrow$ einzelne Recovery-Nachrichten erreichen möglicherweise den eigentlichen Takeover-Knoten nicht

Lösung für das Problem? Wie funktioniert sie?

Lösungsidee

- Aufbauen eines Chord-Ringes (aka Linked List) durch alle Knoten
- Benutzen von Chord-Routing statt (oder in Kombination mit) Greedy Forwarding