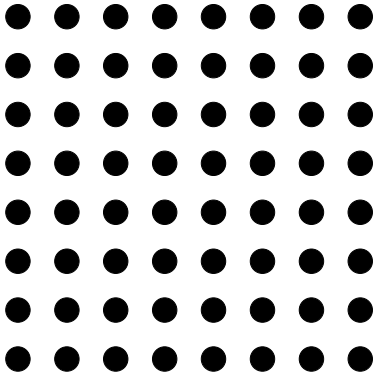
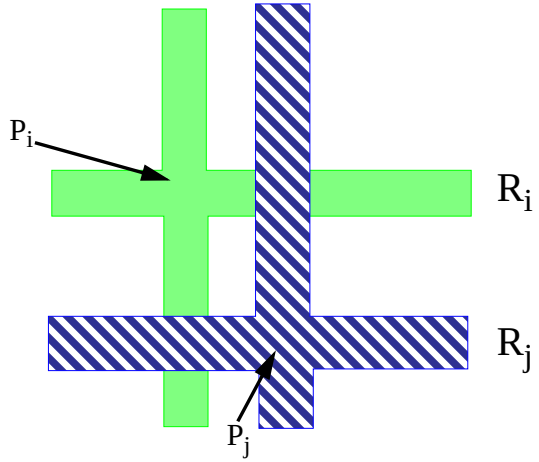
Maekawa's $\sqrt{n}$ -Algorithmus (1985)

"... the algorithm is optimal in terms of the number of messages..."

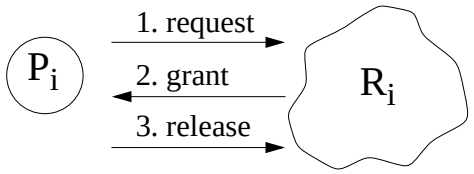
Idee *in etwa:*

- Anordnung der Prozesse in einem $\sqrt{n} \times \sqrt{n}$ - Gitter





- Prozess P_i hat eine Menge von Prozessen R_i, die er (mit request-Nachrichten) *um Erlaubnis fragen* muss
  - hier symbolisiert durch Prozesse in der Spalte / Zeile von P_i
- Die "request-granting" Mengen für je zwei Prozesse *überschneiden* sich garantiert! ($\forall i,j: R_i \cap R_j \neq \emptyset$)
- Grundidee:

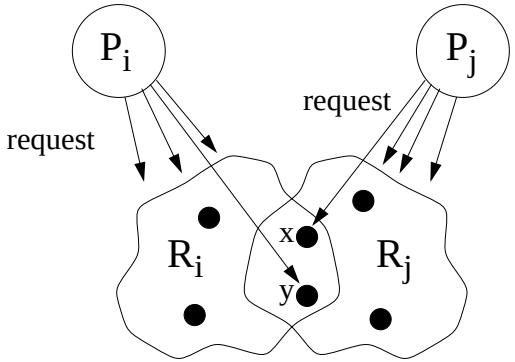


- Nachrichtenkomplexität: $3\,|R_i|$
  --> minimale Mächtigkeit der R_i?

Ein Prozess wartet auf "grant" seiner Menge. Erst dann darf er den kritischen Abschnitt betreten. Nach Verlassen Menge mit "release" informieren.

Eine Erlaubnis ("grant") wird zu einem Zeitpunkt nur einem Bewerber erteilt.

Deadlock-Problematik



Beachte: Zweckmässigerweise ist oft $P_k \in R_k$, falls dies möglich ist.

Im Szenario könnte dann z.B. $x = P_j$ und $y = P_i$ sein.

- y antwortet P_i mit "grant", nicht jedoch P_j
- x antwortet P_j mit "grant", nicht jedoch P_i

==> *Deadlock*, P_i und P_j warten auf weitere Zusage!

Lösung erfordert weitere Nachrichtentypen zur Deadlockvermeidung (bzw. Deadlockbehebung --> Symmetriebrechung)

==> soll hier nicht behandelt werden (--> Literatur)

==> erhöht Nachrichtenkomplexität jedoch nur um konstanten Faktor

---

Ist auch ein Deadlock möglich, wenn $|R_i \cap R_j| = 1$ für alle i,j ?

Ja, siehe nebenstehendes Szenario!

- Prozese i,j und k wenden sich gleichzeitig an ihre entsprechenden Mengen
- jeweils ein Prozess daraus antwortet mit "ja"