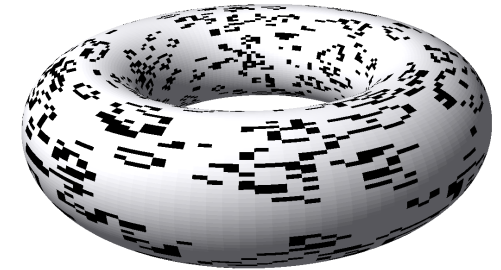


Einführung in die Programmierung

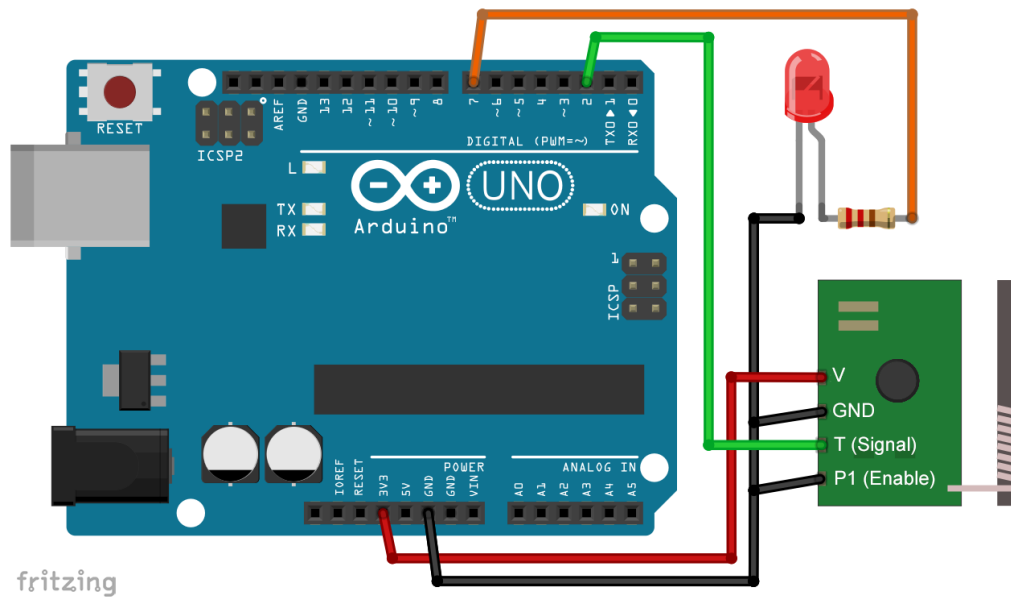
Tutorium 5

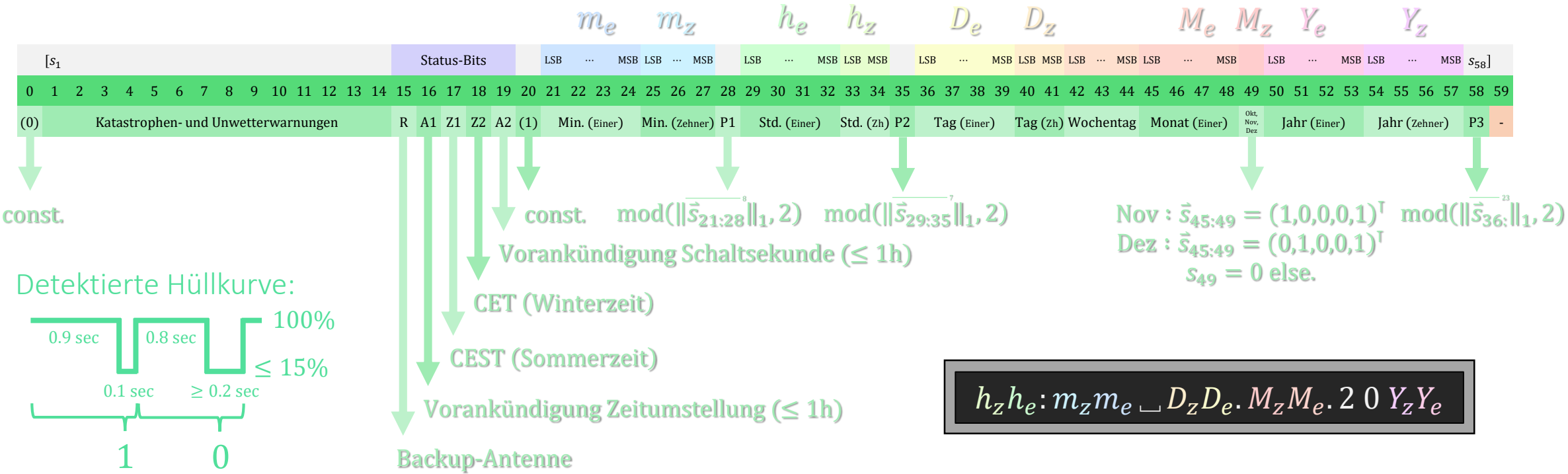


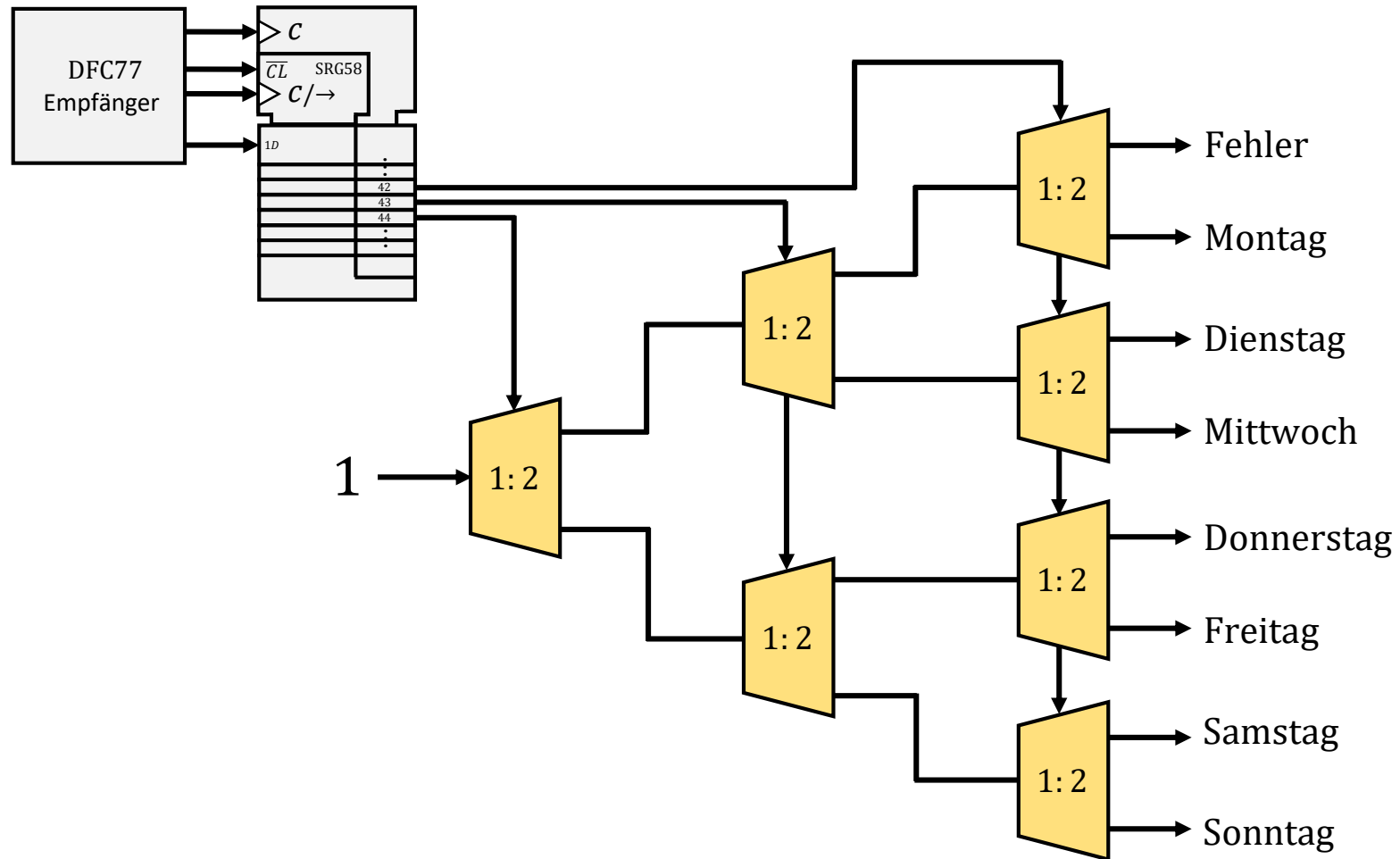
Wintersemester 2025/2026

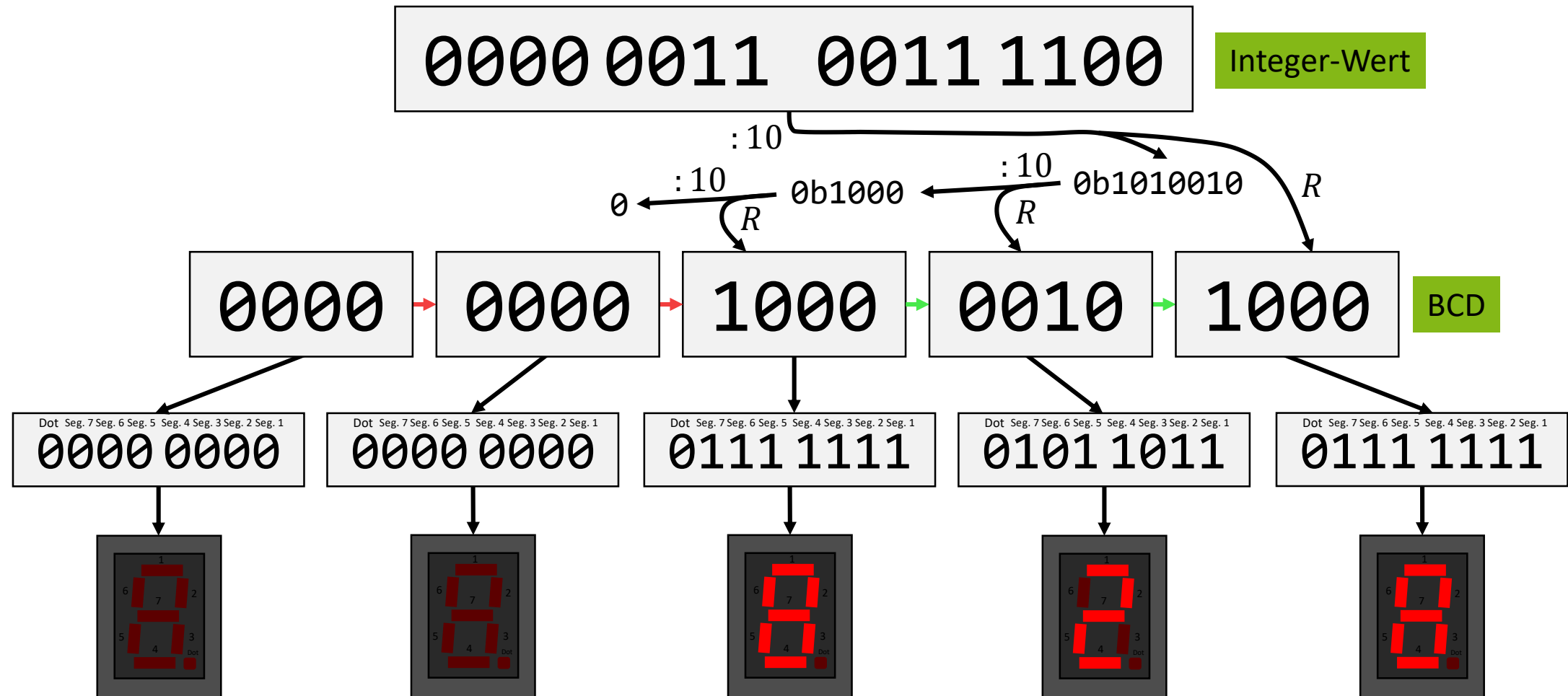
Arbeitsgruppe Systemsoftware
Angewandte Informatik 12



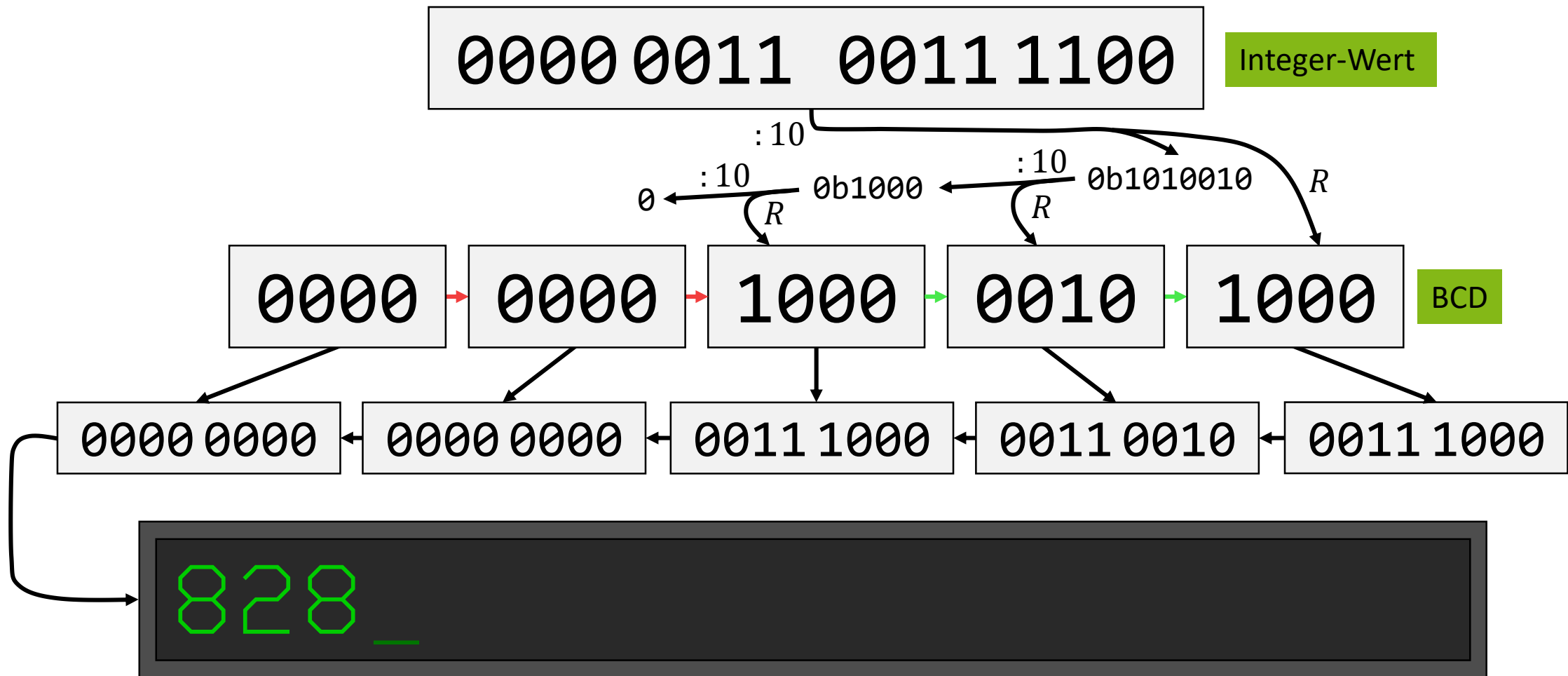


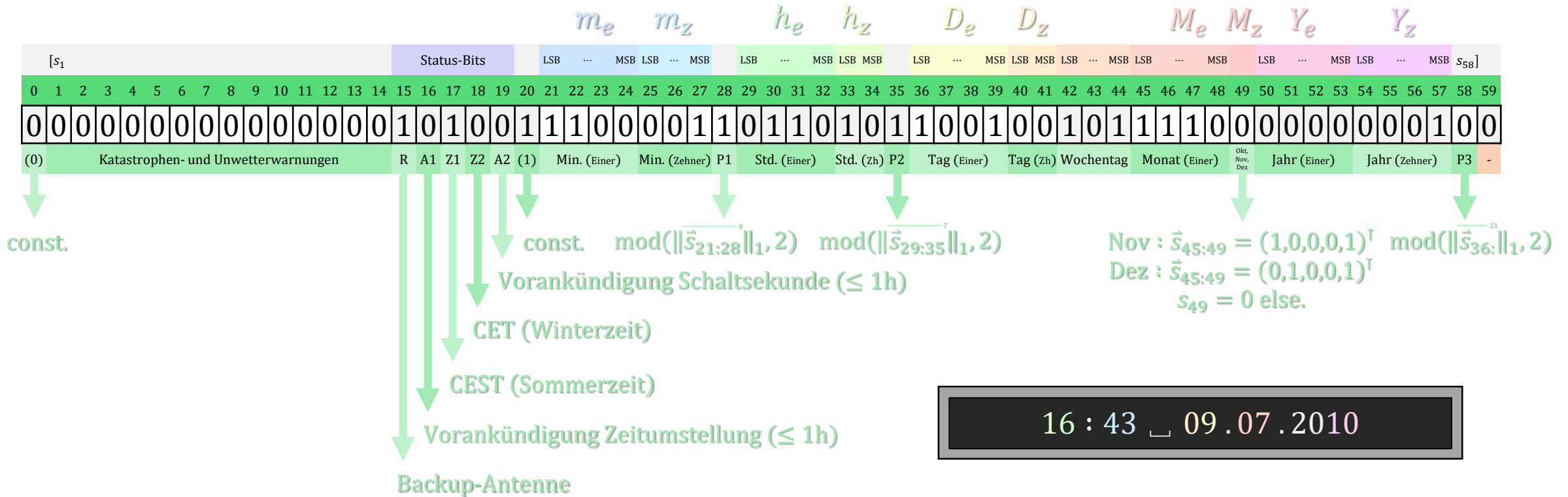






```
printf("%hu", 828);
```





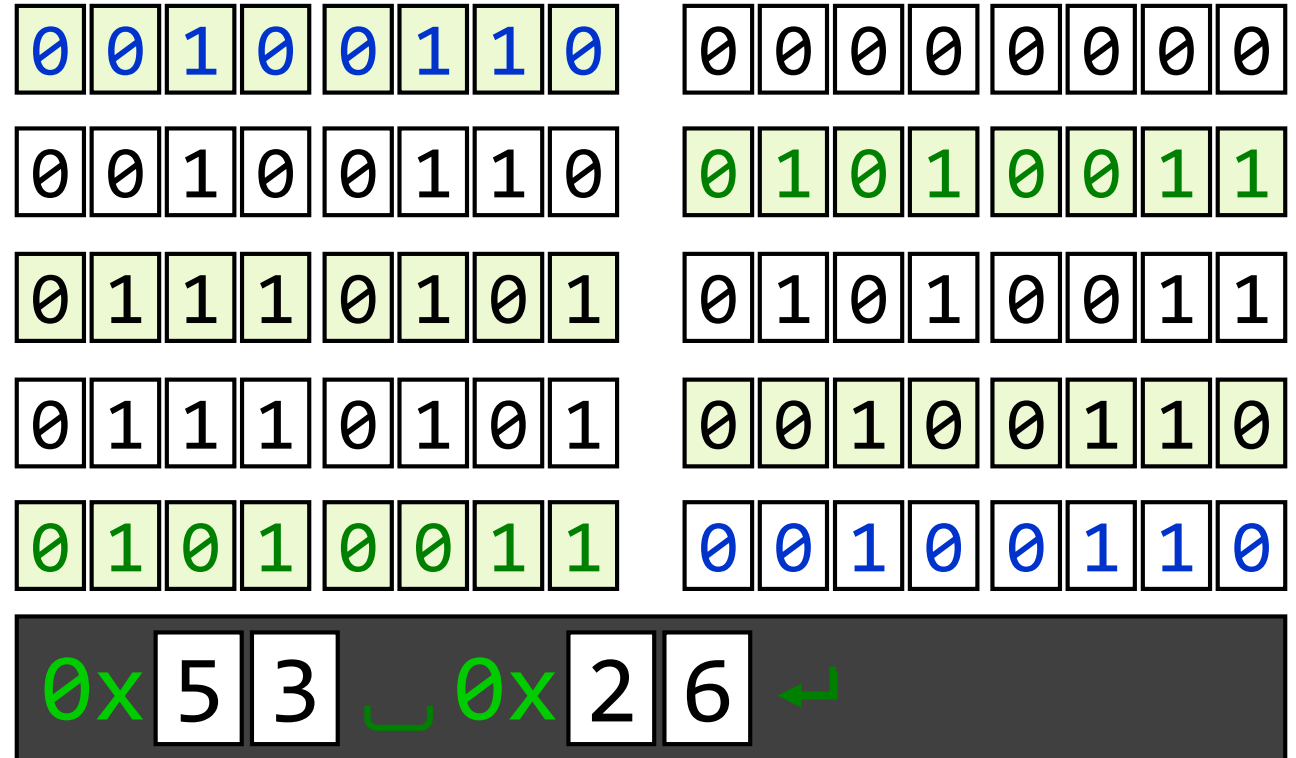

```

uint8_t x = 0b00100110; // 0x26
uint8_t y = 0b01010011; // 0x53

x = x ^ y;
y = x ^ y;
x = x ^ y;

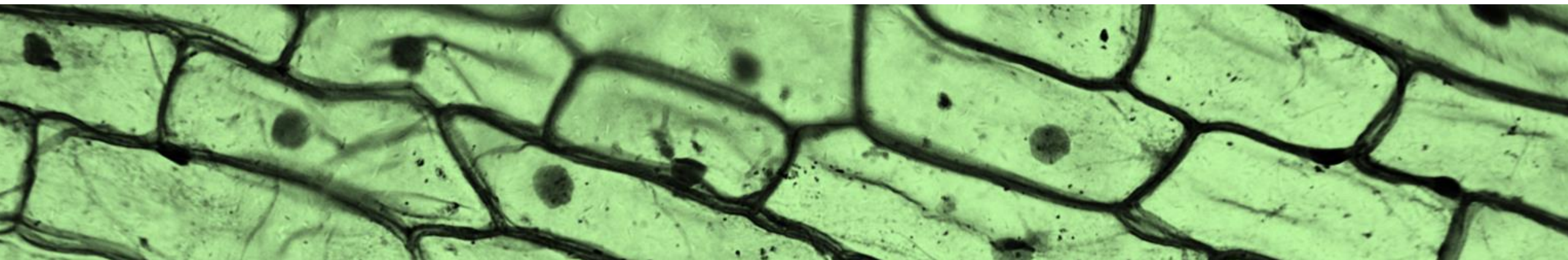
printf("0x%02X 0x%02X\n\r", x, y);

```

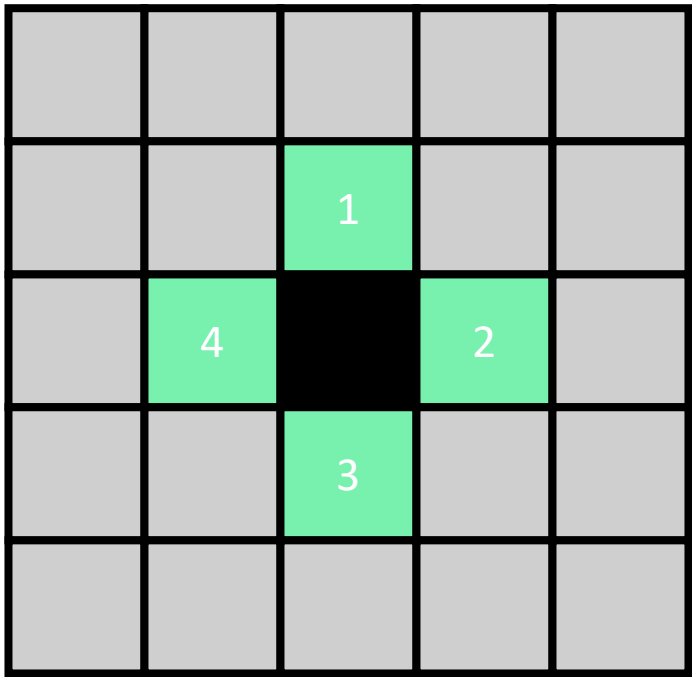


Ein zellulärer Automat $\mathcal{A} := \langle \Omega, Q, \dot{U}_1, \delta \rangle$ setzt sich üblicherweise aus den folgenden vier Komponenten zusammen:

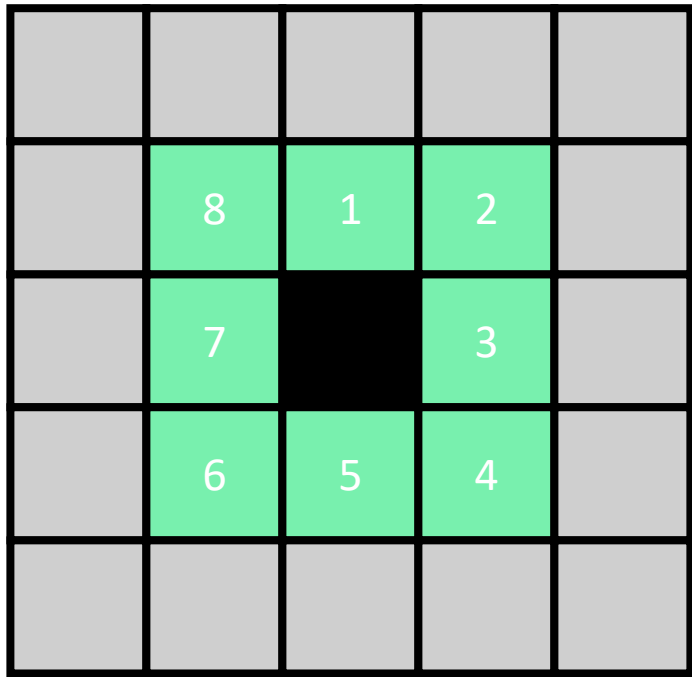
- Ω ist die **Zellmenge** des Automaten; jedes $z \in \Omega$ wird **Zelle** genannt
- Q ist die **Zustandsmenge** des Automaten; $q \in Q$ sind **Zellzustände**
- $\dot{U} : N \times \Omega \rightarrow \mathfrak{P}(\Omega)$ ist die **Nachbarschaftsrelation** des Automaten
- $\delta : Q \times \Omega^{[0..n]} \rightarrow Q$ ist die **Zustandsübergangsfunktion** des Automaten



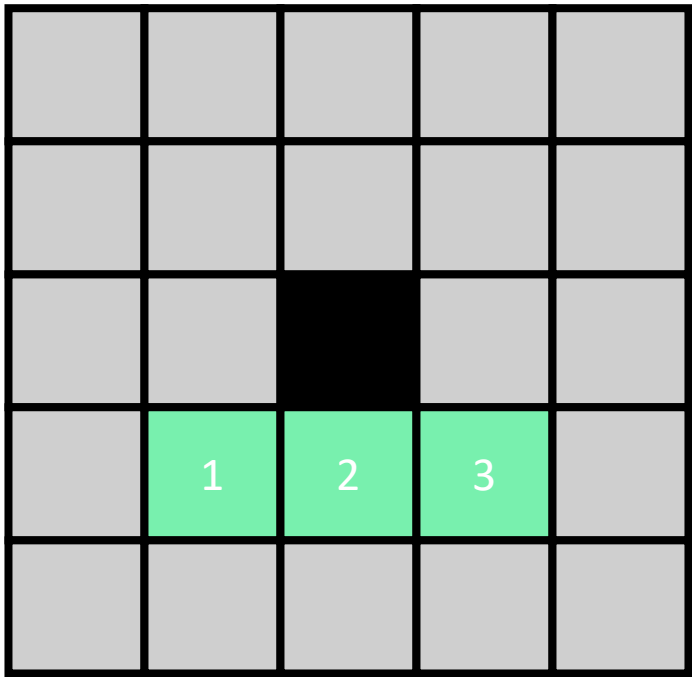
Von Neumann-Nachbarschaft



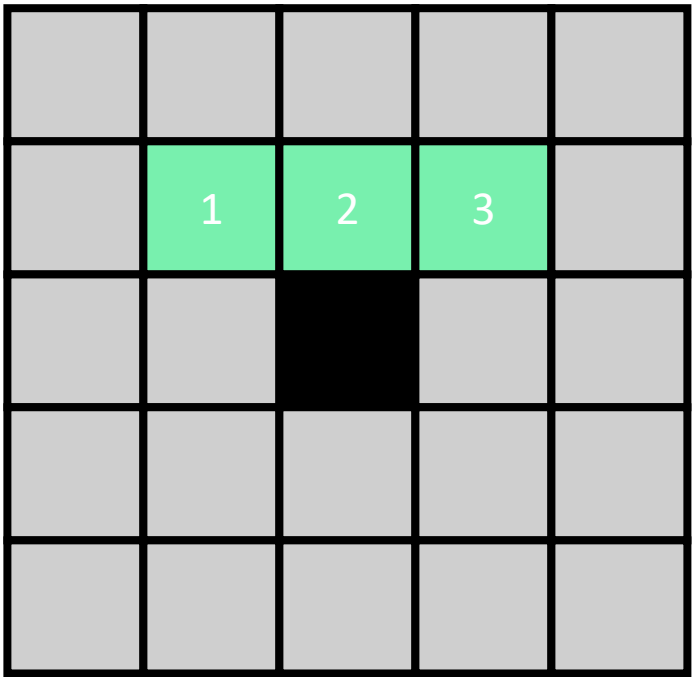
Moore-Nachbarschaft



Bottom-Up-Nachbarschaft \dot{U}_{BU}



Top-Down-Nachbarschaft \dot{U}_{TD}

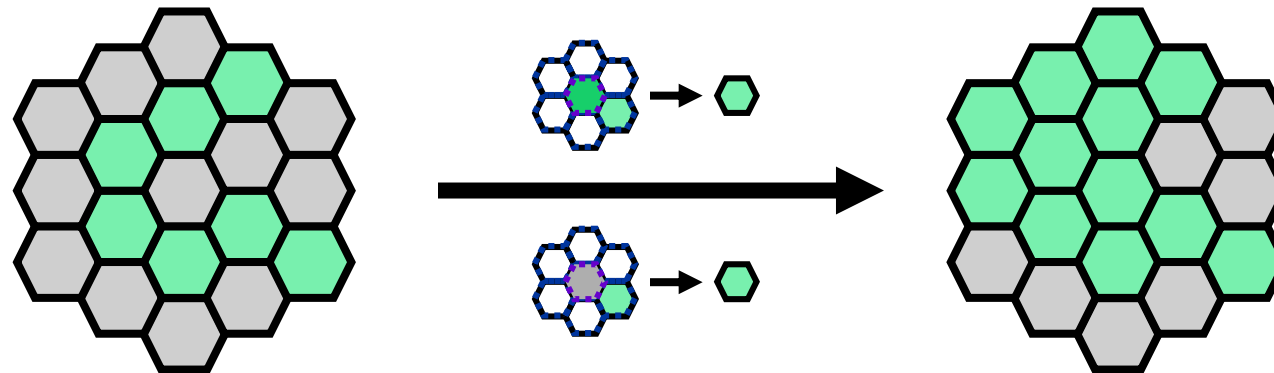


- Eine **globale Konfiguration** kann als Abbildung $q : \Omega \rightarrow Q$ von der Menge der Zellen Ω auf Q verstanden werden
- Globale Konfigurationen beschreiben den aktuellen Status aller Zellen
- Eine globale Folgekonfiguration wird als **Generation** bezeichnet
- \mathcal{A} wird zur Initialisierung mit einer globalen Start-konfiguration S , dem sogenannten **Garten Eden** versehen



Die **globale Übergangsfunktion** $\Delta : Q^\Omega \rightarrow Q^\Omega$ ordnet jeder globalen Konfiguration des zellulären Automaten eine **Nachfolgekongfiguration** zu:

$$q \mapsto q_{\text{neu}} : \Omega \rightarrow Q; \quad z \mapsto \delta(q(z), \mu(\dot{U}_1(z)))$$




```
bool rule[16] = {0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0};

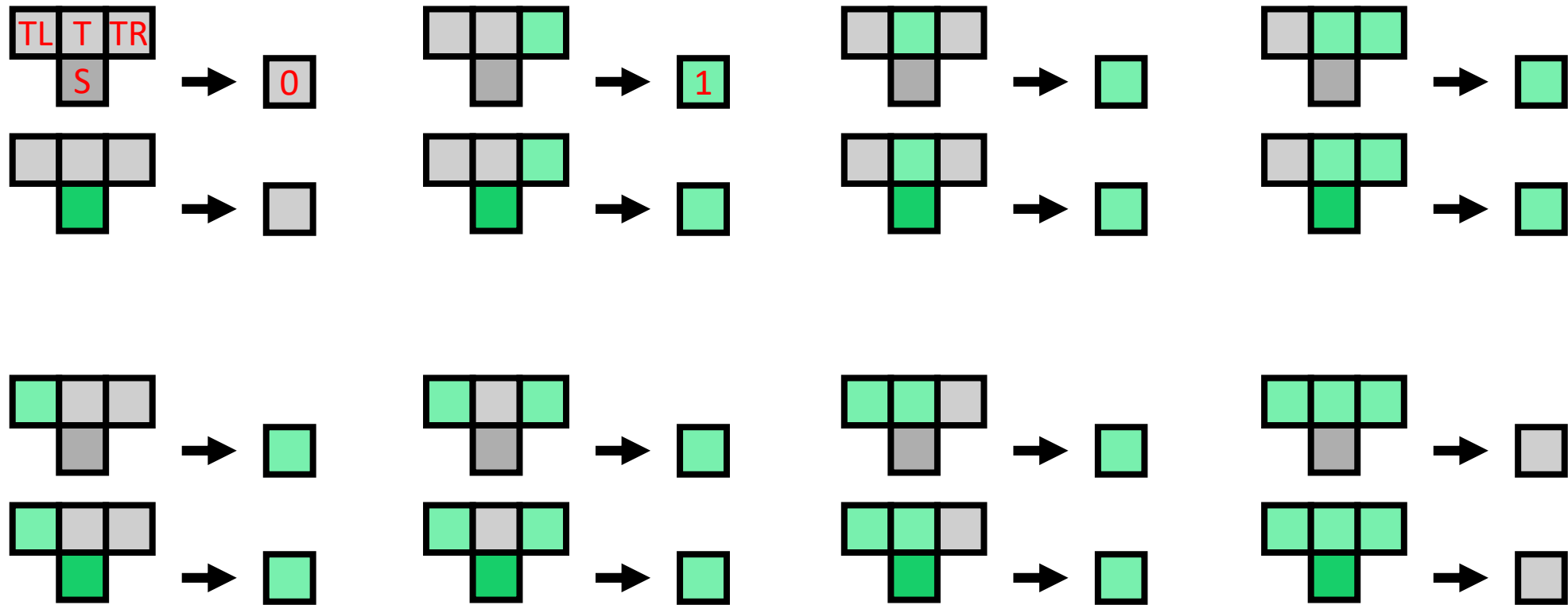
void update_grid(double currentTime) {
    bool new_grid[GRID_SIZE][GRID_SIZE] = {0};

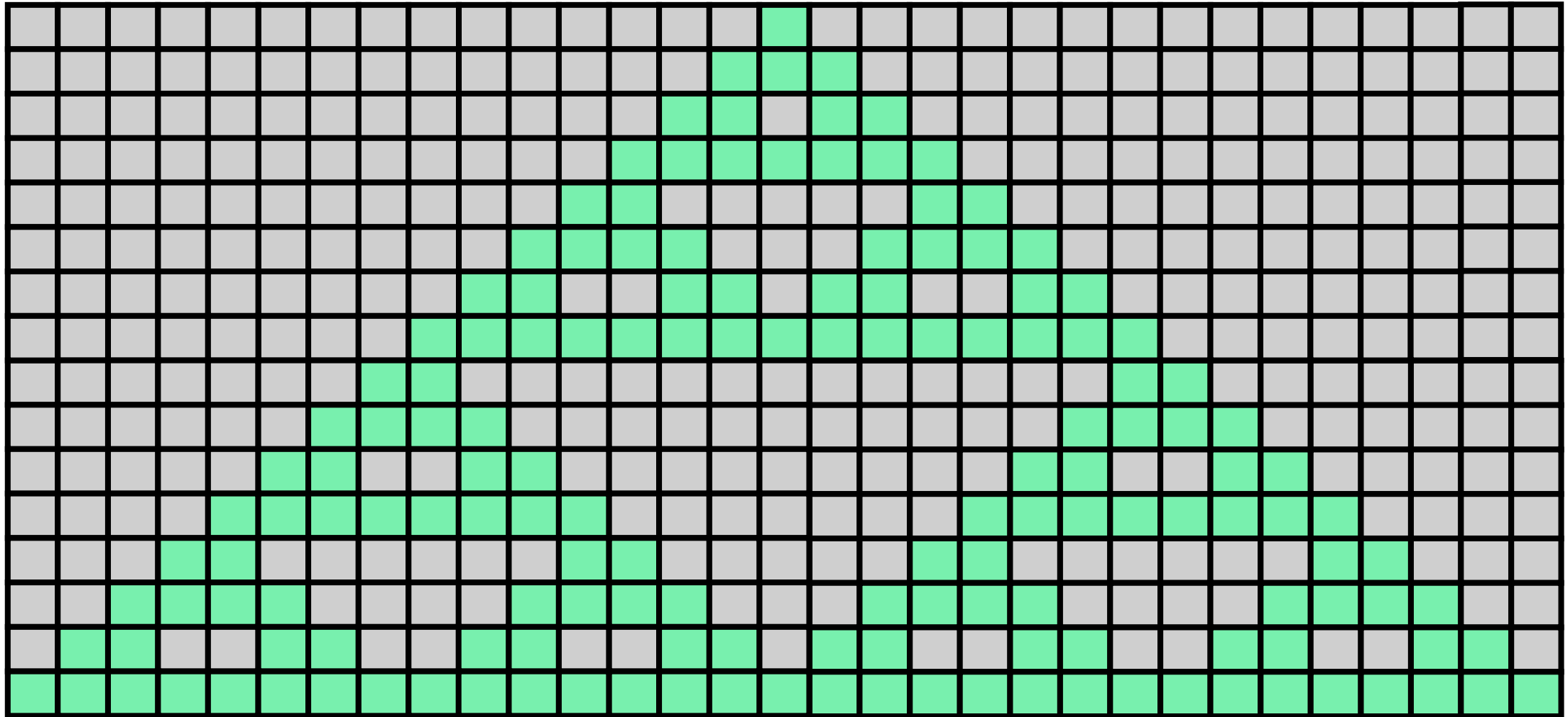
    for (int y = 0; y < GRID_SIZE; ++y) {
        for (int x = 0; x < GRID_SIZE; ++x) {
            bool S = grid[x][y];
            bool TL = (y+1 < GRID_SIZE && x < GRID_SIZE-1) ? grid[x+1][y+1] : false;
            bool T = (y+1 < GRID_SIZE) ? grid[x][y+1] : false;
            bool TR = (y+1 < GRID_SIZE && x > 0) ? grid[x-1][y+1] : false;
            int idx = (TL << 3) | (T << 2) | (TR << 1) | S;
            bool next = rule[idx];
            next = next || S;
            new_grid[x][y] = next;
        }
    }

    memcpy(grid, new_grid, sizeof(grid));
    running = (glfwGetKey(glfwGetCurrentContext(), GLFW_KEY_N) != GLFW_PRESS);
    lastUpdateTime = currentTime;
}
```

$$\mathcal{A} := \langle \Omega := [0 \dots 31] \times [0 \dots 31], Q := \{0,1\}, \dot{U}_{TD}, \delta \rangle$$

δ :

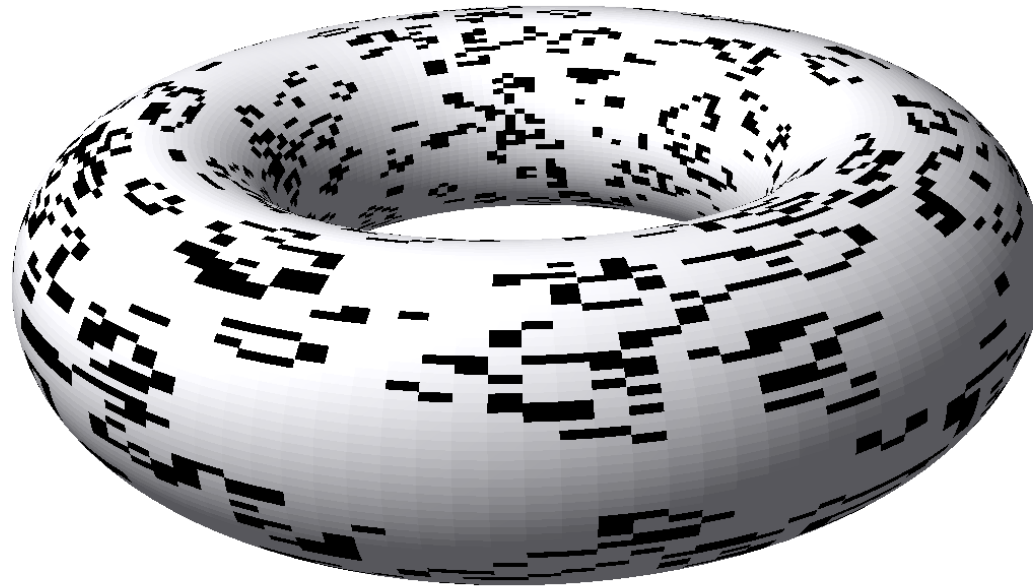


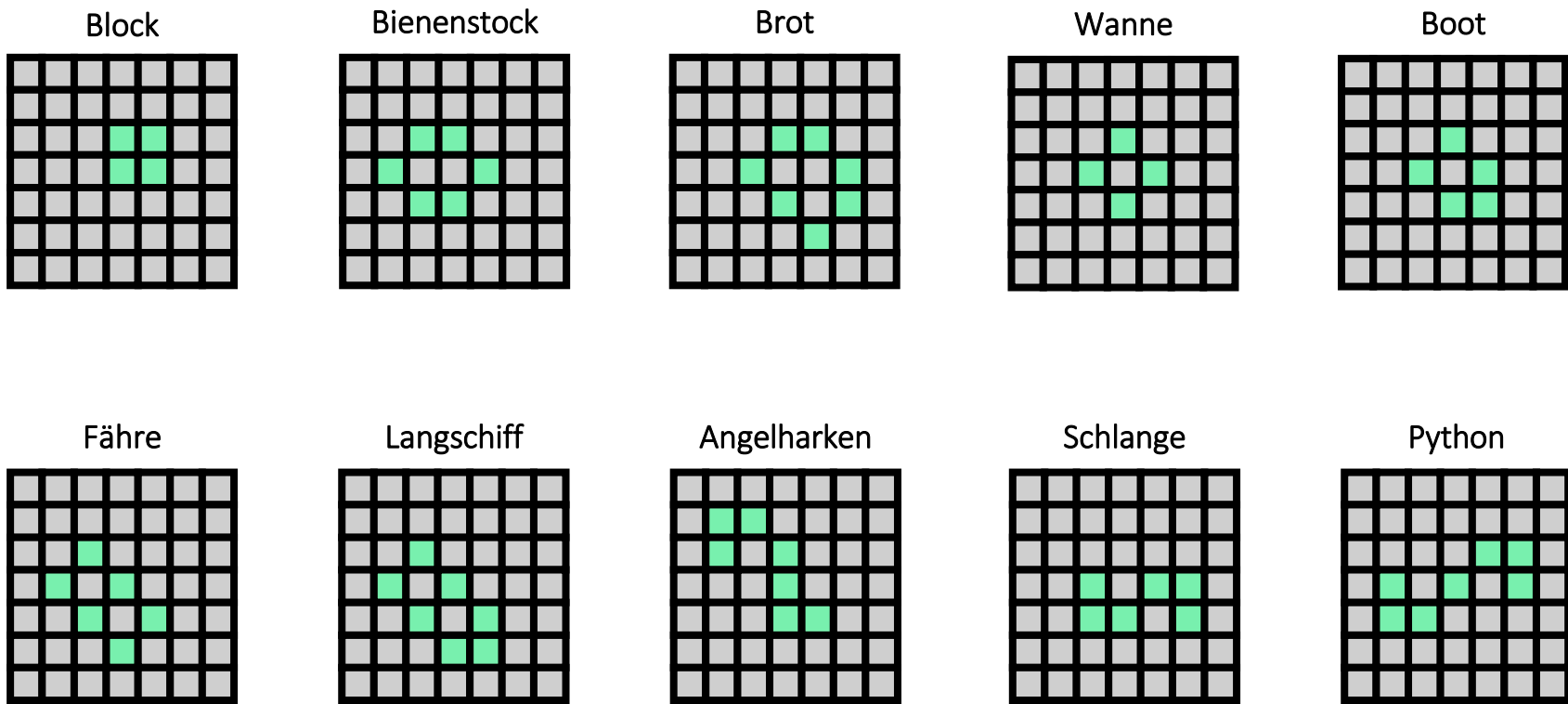


- Ein bekanntes Beispiel für zelluläre Automaten $\mathcal{A} := \langle \Omega, Q, \dot{U}_1, \delta \rangle \dots$
 - ... mit $\Omega := \mathbb{Z}^2$ und Moore-Nachbarschaften
 - ... mit den Zuständen $Q := \{0,1\} \rightarrow 1 \hat{=} \text{'lebendig'}$ und $0 \hat{=} \text{'tot'}$

ist **Conways Spiel des Leben** und umfasst folgende Übergangsregeln:

- Eine lebende Zelle stirbt, wenn sie weniger als zwei lebendige Nachbarn hat
 - Eine lebende Zelle mit zwei oder drei lebendige Nachbarn lebt weiter
 - Eine lebende Zelle mit mehr als drei lebenden Nachbarn stirbt
 - Ein freier Platz bzw. eine tote Zelle wird mit einer lebendigen Zelle ersetzt, wenn sie genau drei lebende Nachbarzellen hat
- Conways Spiel des Leben ist in beliebiger Topologie simulierbar
 - Torus (periodische Randbedingungen)





Kok's Galaxie

