Übung 11 – Regions and Zones

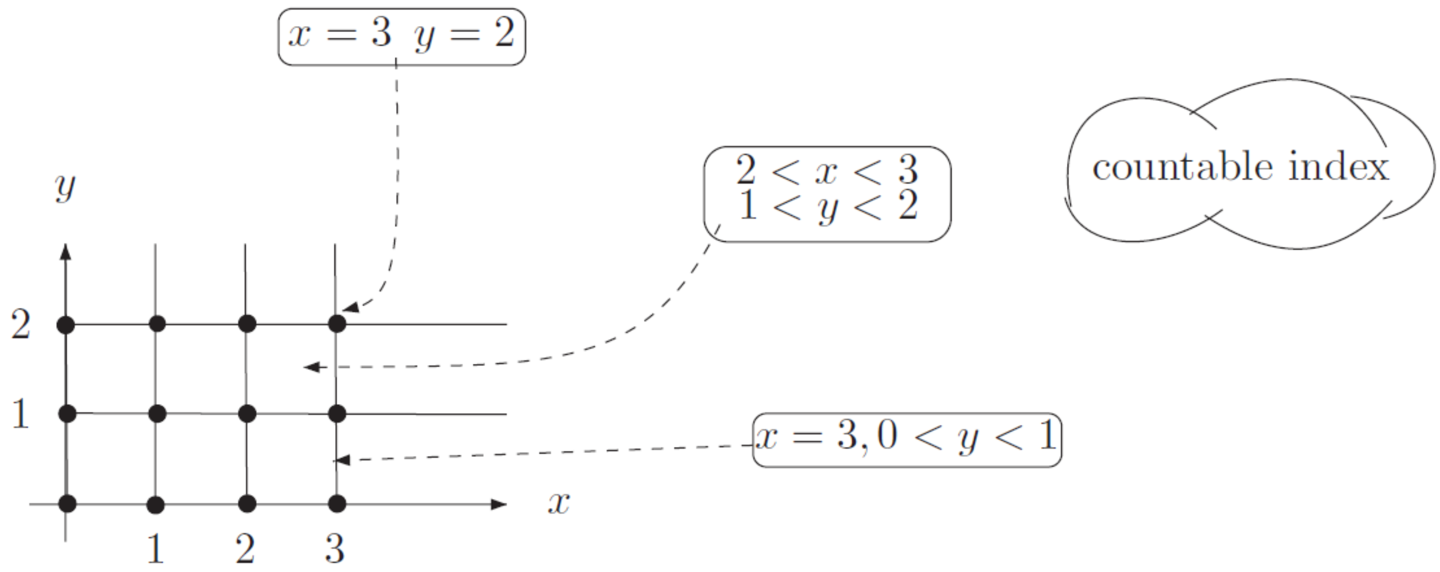
Christian Kroiß

- Clock equivalence and Regions revisited
- Operationen an Difference Bound Matrices

1. Schritt: Da Uhren nur mit ganzzahligen Werten verglichen werden, reicht:

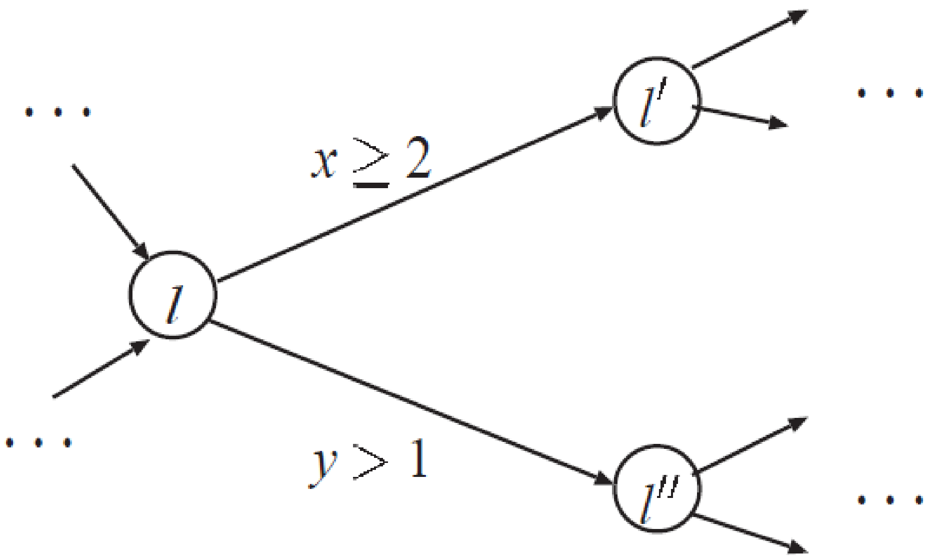
$$\eta \cong_1 \eta' \Leftrightarrow$$

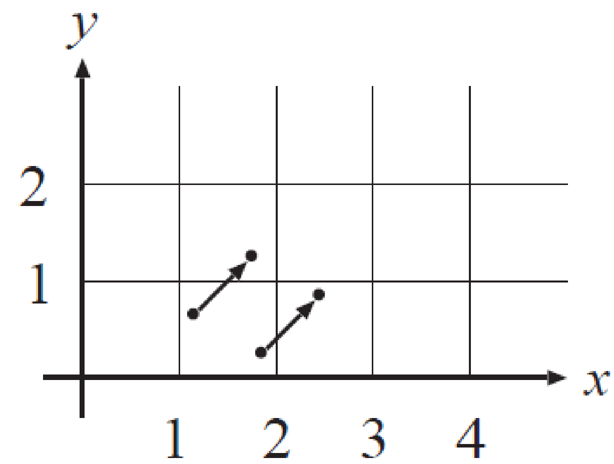
$$\lfloor \eta(x) \rfloor \;=\; \lfloor \eta'(x) \rfloor \quad \text{and} \quad \mathit{frac}(\eta(x)) = 0 \text{ iff } \mathit{frac}(\eta'(x)) = 0.$$



Equivalence classes:

- the corner points (q, p)
- the line segments $\{ (q, y) \mid p < y < p{+}1 \}$ and $\{ (x, p) \mid q < x < q{+}1 \}$, and
- the content of the squares $\{ (x, y) \mid q < x < q{+}1 \ \wedge \ p < y < p{+}1 \}$

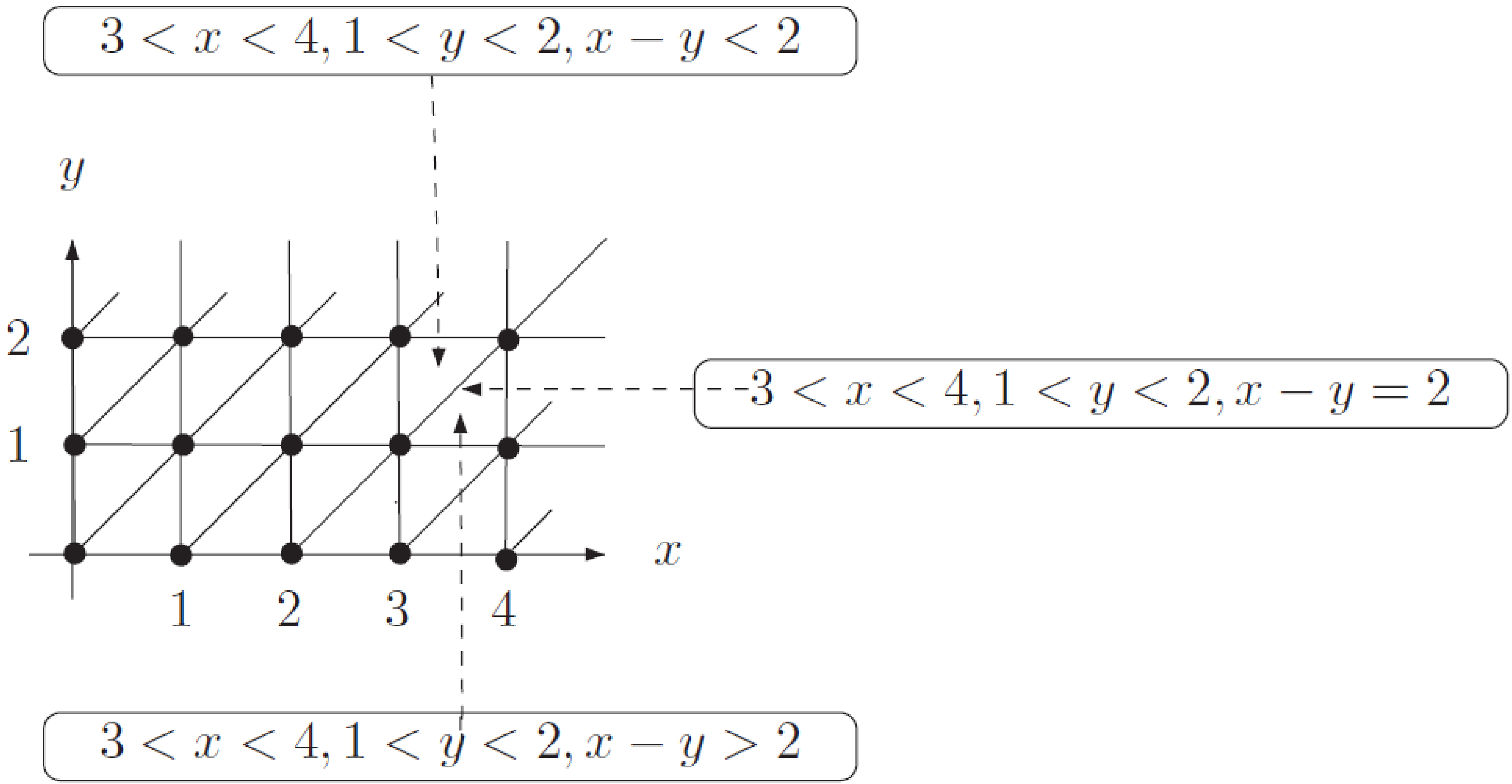




If $frac(\eta(x)) < frac(\eta(y))$, then β is enabled before α;
if $frac(\eta(x)) > frac(\eta(y))$, action α is enabled first.

Partitioning is too coarse!

$$frac(\eta(x)) \leq frac(\eta(y)) \text{ if and only if } frac(\eta'(x)) \leq frac'(\eta(y))$$



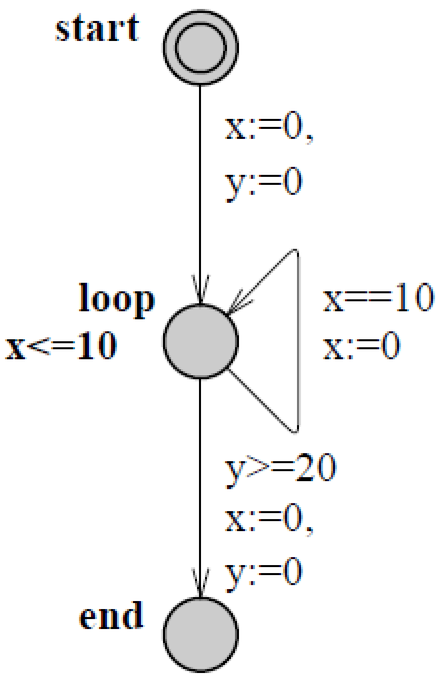
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

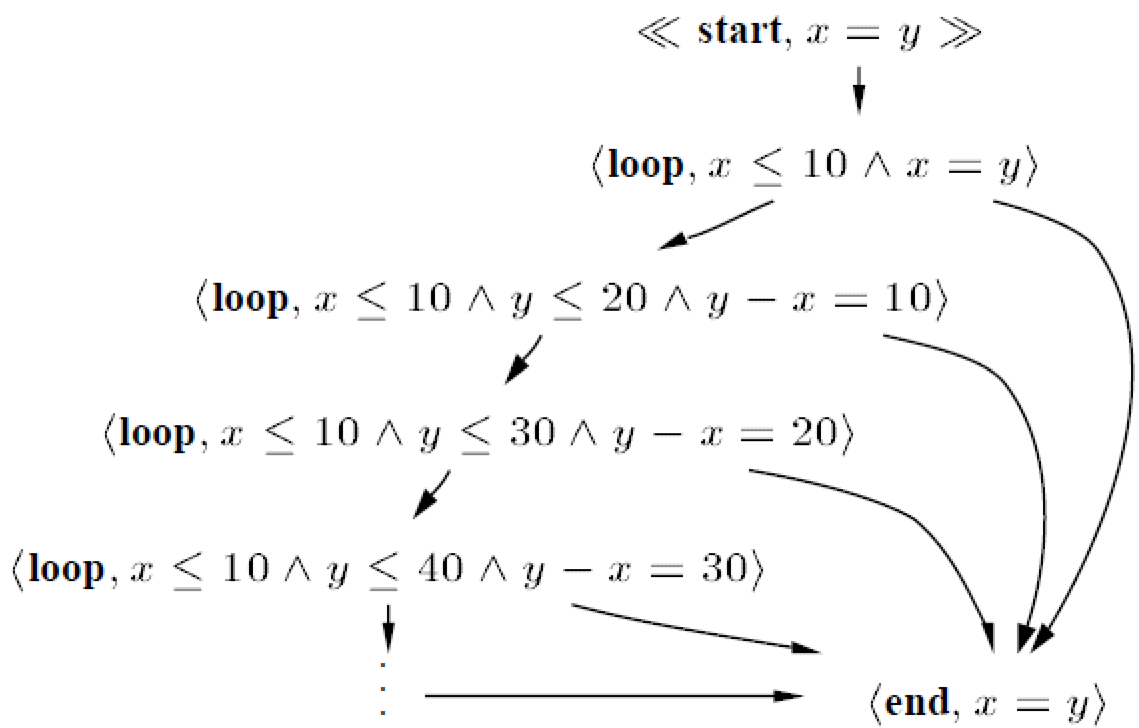
Definition: clock equivalence

Let TA be a timed automaton, Φ a $\text{TCTL}_\Diamond$ formula (both over set C of clocks), and c_x the largest constant with which $x \in C$ is compared with in either TA or Φ. Clock valuations $\eta, \eta' \in Eval(C)$ are *clock-equivalent*, denoted $\eta \cong \eta'$ if and only if either

- for any $x \in C$ it holds that $\eta(x) > c_x$ and $\eta'(x) > c_x$, or
- for any $x, y \in C$ with $\eta(x), \eta'(x) \leqslant c_x$ and $\eta(y), \eta'(y) \leqslant c_y$ all the following conditions hold:
  - $\lfloor \eta(x) \rfloor = \lfloor \eta'(x) \rfloor$ and $frac(\eta(x)) = 0$ iff $frac(\eta'(x)) = 0$,
  - $frac(\eta(x)) \leqslant frac(\eta(y))$ iff $frac(\eta'(x)) \leqslant frac(\eta'(y))$.

Zone Graph kann unendlich werden





Lösung: normalisieren (hier wenn Automat nur Vergleiche mit Konstanten enthält)

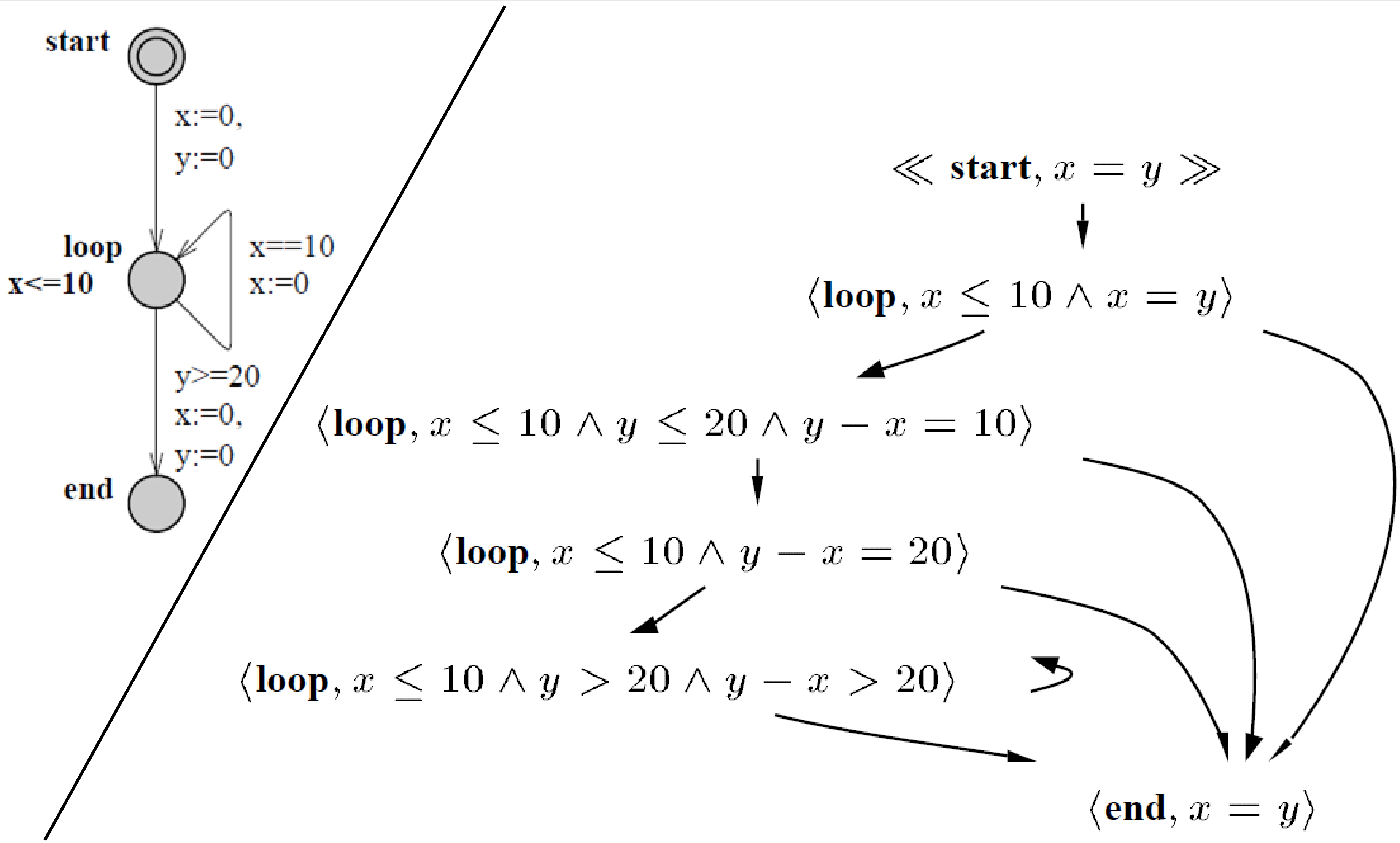
Definition 8 (k-Normalization) *Let D be a zone and k a clock ceiling. The semantics of the k-normalization operation on zones is defined as follows:*

$$\mathsf{norm}_k(D) = \{u \mid u \dot{\sim}_k v, v \in D\}$$

Note that the normalization operation is indexed by a clock ceiling k. According to [Rok93,Pet99], $\mathsf{norm}_k(D)$ can be computed from the canonical representation of D by

1. removing all constraints of the form $x < m$, $x \leq m$, $x - y < m$ and $x - y \leq m$ where $m > k(x)$,
2. replacing all constraints of the form $x > m$, $x \geq m$, $x - y > m$ and $x - y \geq m$ where $m > k(x)$ with $x > k(x)$ and $x - y > k(x)$ respectively.

Zone-Normalisierung



Konstruktion von DBMs:

As an example, consider the zone $D = x - \mathbf{0} < 20 \wedge y - \mathbf{0} \leq 20 \wedge y - x \leq 10 \wedge x - y \leq -10 \wedge \mathbf{0} - z < 5$. To construct the matrix representation of D, we number the clocks in the order $\mathbf{0}, x, y, z$. The resulting matrix representation is shown below:

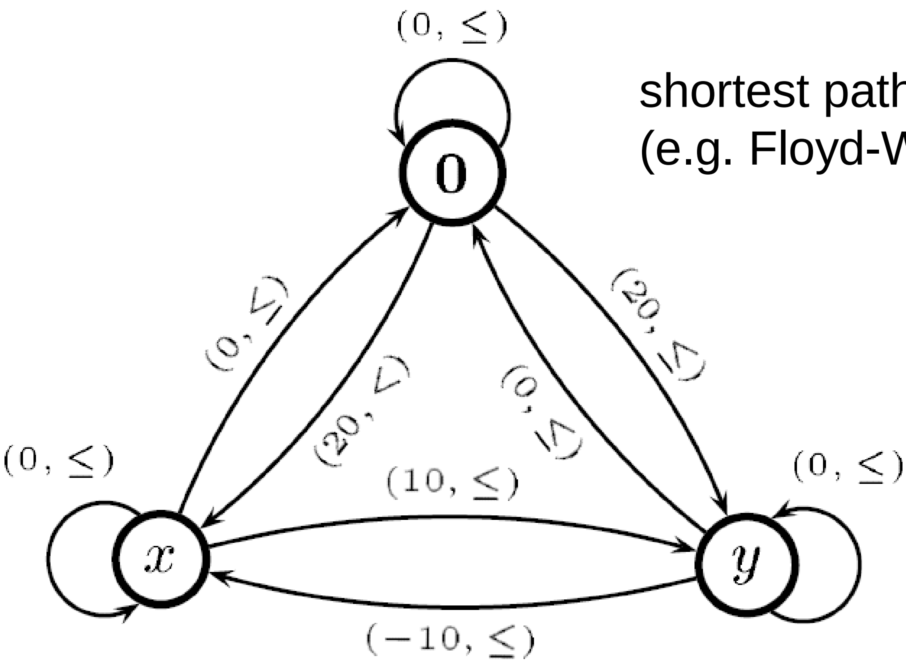
$$M(D) = \begin{pmatrix} (0\,,\leq) & (0\,,\leq) & (0\,,\leq) & (5\,,<) \\ (20\,,<) & (0\,,\leq) & (-10\,,\leq) & \infty \\ (20\,,\leq) & (10\,,\leq) & (0\,,\leq) & \infty \\ \infty & \infty & \infty & (0\,,\leq) \end{pmatrix}$$

To manipulate DBMs efficiently we need two operations on bounds: comparison and addition. We define that $(n, \preceq) < \infty$, $(n_1, \preceq_1) < (n_2, \preceq_2)$ if $n_1 < n_2$ and $(n, <) < (n, \leq)$. Further we define addition as $b_1 + \infty = \infty$, $(m, \leq) + (n, \leq) = (m + n, \leq)$ and $(m, <) + (n, \preceq) = (m + n, <)$.

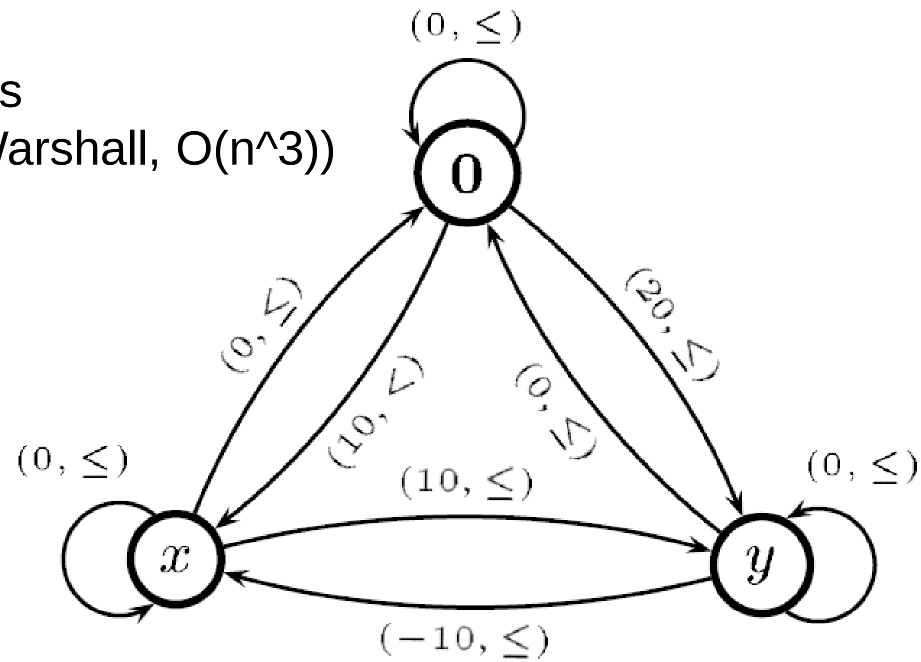
Kanonische DBMs

Canonical DBMs Usually there are an infinite number of zones sharing the same solution set. However, for each family of zones with the same solution set there is a unique constraint where no atomic constraint can be strengthened without losing solutions.

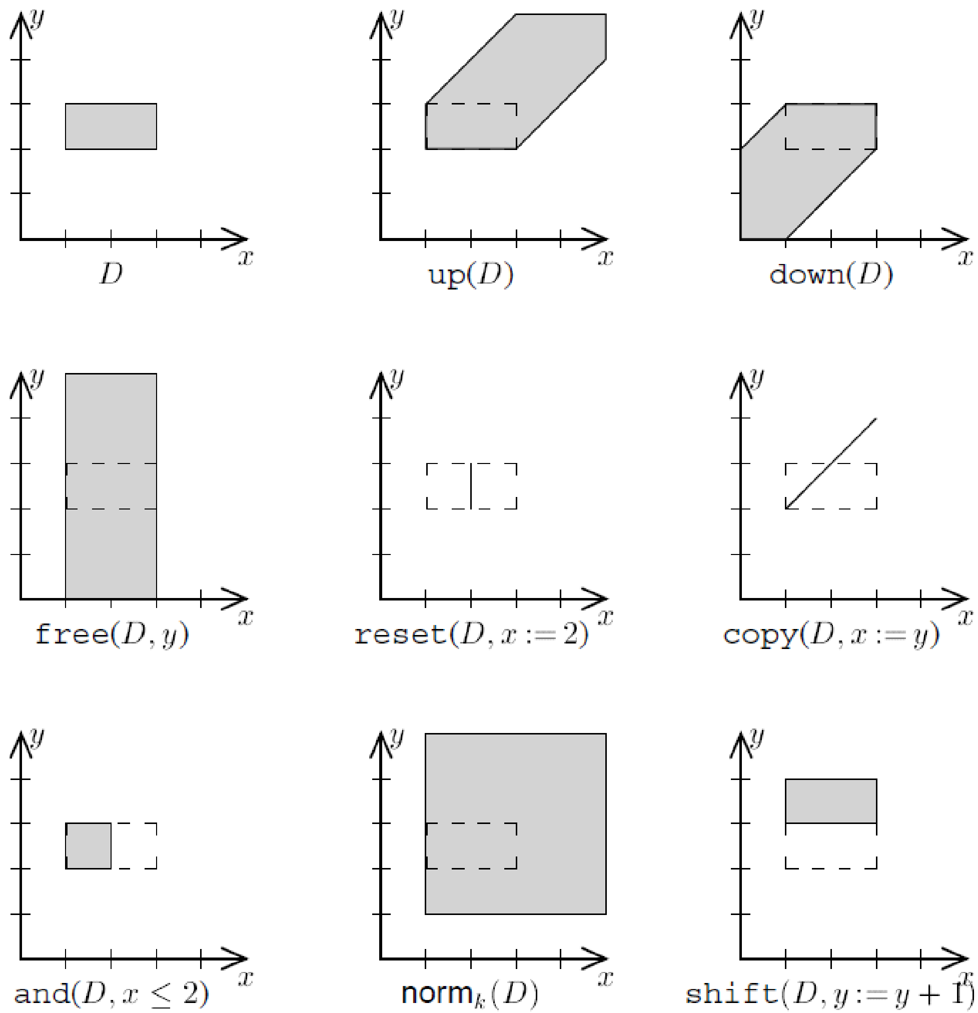
➔ Derive tightest constraint on each clock difference



shortest paths
(e.g. Floyd-Warshall, O(n^3))



Operationen auf DBMs



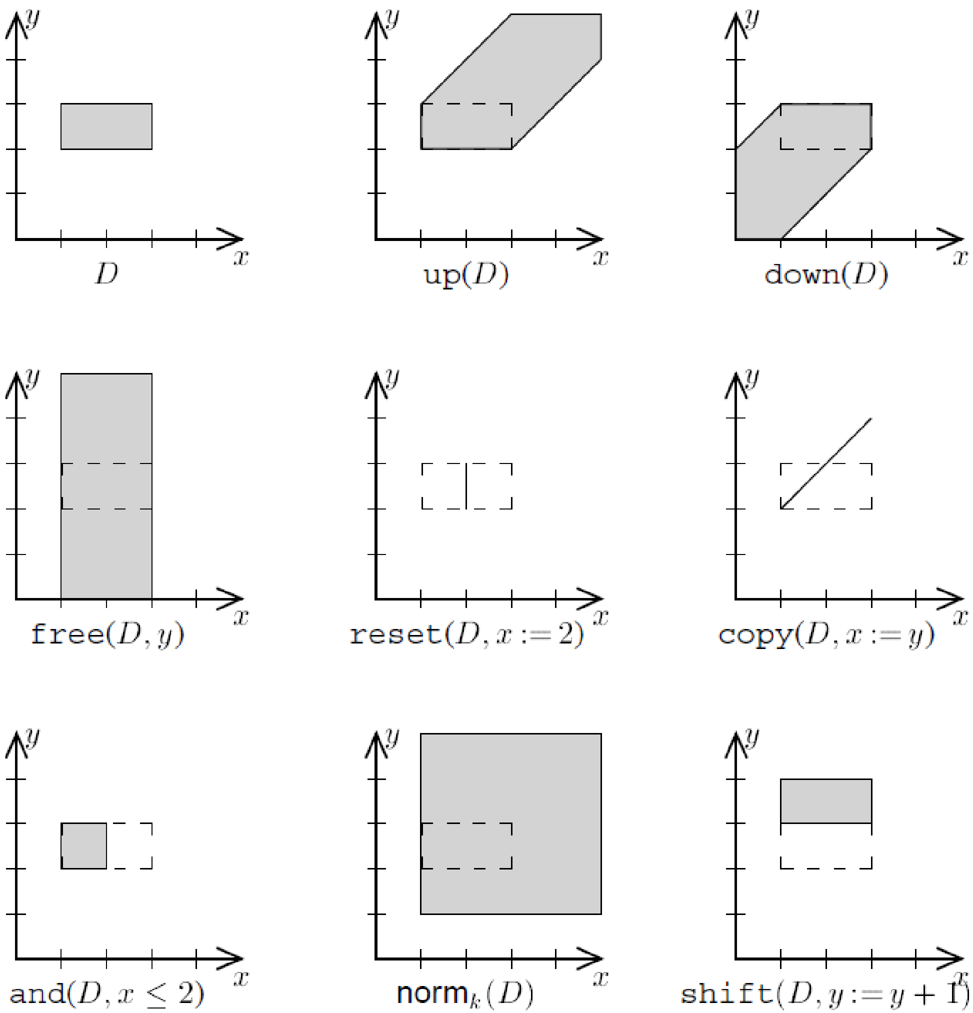
Algorithm 6 up(D)

for $i := 1$ **to** n **do**
  $D_{i0} := \infty$
end for

Algorithm 10 reset($D, x := m$

for $i := 0$ **to** n **do**
  $D_{xi} := (m, \leq) + D_{0i}$
  $D_{ix} := D_{i0} + (-m, \leq)$
end for

Operationen auf DBMs



Algorithm 8 and(D, g)

if $D_{yx} + (m, \preceq) < 0$ **then**
  $D_{00} = (-1, \leq)$
else if $(m, \preceq) < D_{xy}$ **then**
  $D_{xy} = (m, \preceq)$
  for $i := 0$ **to** n **do**
    for $j := 0$ to n **do**
      if $D_{ix} + D_{xj} < D_{ij}$ **then**
        $D_{ij} = D_{ix} + D_{xj}$
      end if
      if $D_{iy} + D_{yj} < D_{ij}$ **then**
        $D_{ij} = D_{iy} + D_{yj}$
      end if
    end for
  end for
end if

Algorithm 1 Reachability analysis

$\text{PASSED} = \emptyset, \text{WAIT} = \{\langle l_0, D_0 \rangle\}$
while $\text{WAIT} \neq \emptyset$ **do**
  take $\langle l, D \rangle$ from WAIT
  if $l = l_f \wedge D \cap \phi_f \neq \emptyset$ **then return** "YES"
  if $D \not\subseteq D'$ for all $\langle l, D' \rangle \in \text{PASSED}$ **then**
    add $\langle l, D \rangle$ to PASSED
    for all $\langle l', D' \rangle$ such that $\langle l, D \rangle \rightsquigarrow k, \mathcal{G}\langle l', D' \rangle$ **do**
      add $\langle l', D' \rangle$ to WAIT
    end for
  end if
end while
return "NO"

Principles of Model Checking

Christel Baier, Joost-Pieter Katoen

The MIT Press (3. Juni 2008)

Sprache: Englisch

ISBN-10: 026202649X

ISBN-13: 978-0262026499