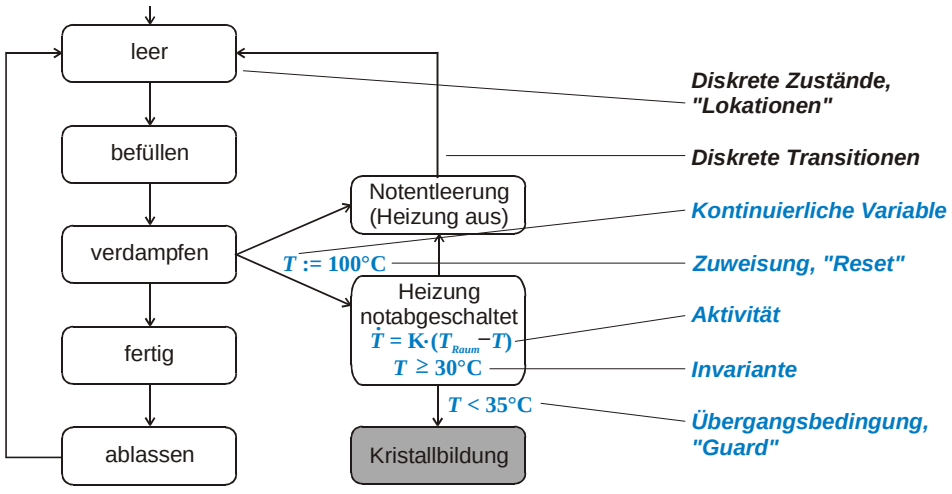
Formale Methoden für eingebettete Systeme
Teil 10: Hybride Systeme

Prof. Dr. Stefan Kowalewski

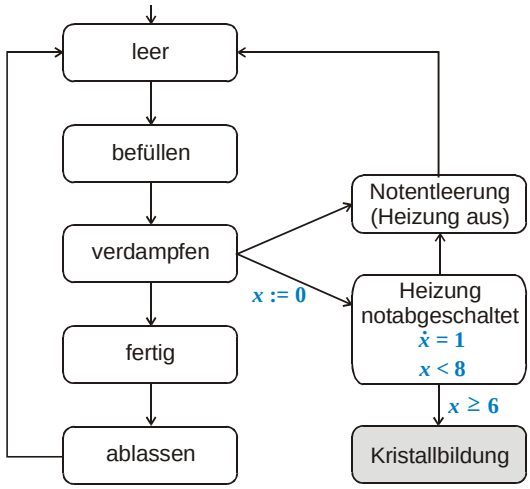
Lehrstuhl Informatik 11
RWTH Aachen

Sommersemester 2005

Reminder: Hybrider Automat
(Henzinger, Alur, Sifakis, 1992)



Echtzeitautomat (Alur, Dill, 1990)



Erreichbarkeitsanalyse

Gegeben:

- ein hybrider Automat (HA) mit einem hybriden Anfangszustand (= diskrete Lokation + kontinuierliche Region)
- ein hybrider Zielzustand.

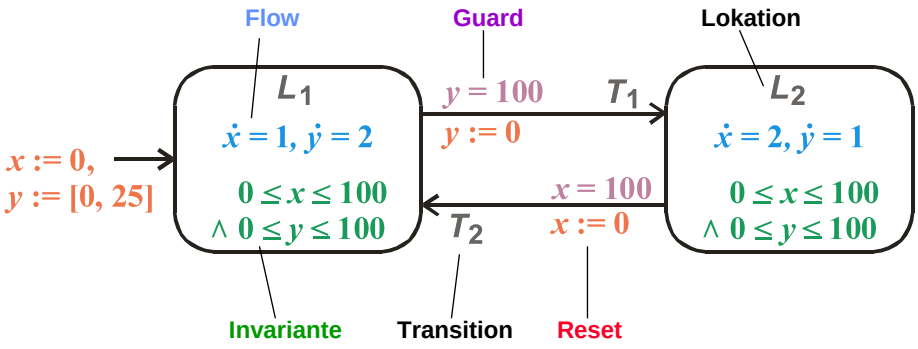
Frage:

Existiert eine Trajektorie (ein „Lauf“ des HA), die vom Anfangs- zum Zielzustand führt?

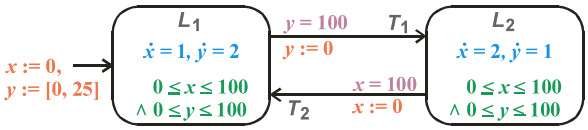
Werkzeuge:

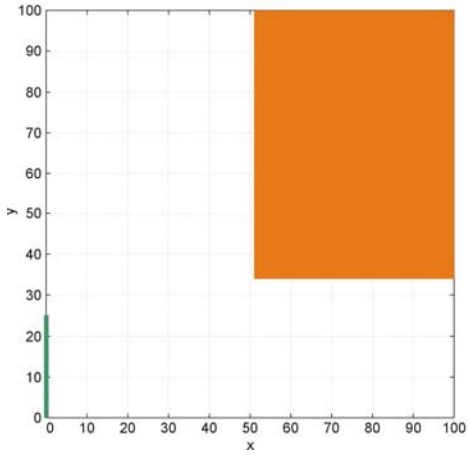
- ***Kronos*** (Verimag, Grenoble), ***Uppaal*** (Uppsala & Aalborg): *Echtzeitautomaten* ($\dot{x} = 1$)
- ***ältere Version von Uppaal:*** *Integrator-Automaten* ($\dot{x} \in \{0, 1\}$)
- ***Hytech*** (Berkeley): „*Lineare“ hybride Automaten* ($\dot{x} \in [\dot{x}_{min}, \dot{x}_{max}]$)

Beispiel



Beispiel (Forts.)





Frage:
Gibt es eine Trajektorie von
(L1, $x = 0 \wedge y \in [0, 25]$)
nach
(*, $x \ge 51 \wedge y \ge 34$)?

Erreichbarkeitsalgorithmus

0. Initialisiere den HA: *L* := Anfangslokation, *P* := Anfangspolyeder
1. Schneide *P* mit der Invariante von *L*.
2. Lasse *P* entsprechend der Aktivität von *L* wachsen.
3. Schneide *P* mit der Invariante von *L*.
4. Stop, falls *L* schon mit *P* besucht wurde. Falls nicht:

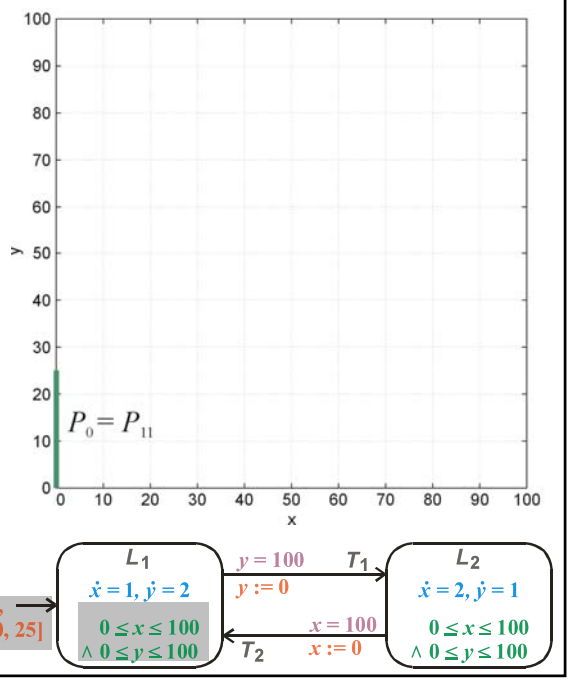
Für alle Transitionen*T* von *L*:

5. Schneide *P* mit dem Guard von *T*. Falls leer, nehme nächste Transition.
6. Falls nicht, setze *P* entspr. Reset-Ausdruck zurück.
7. Setze *L* := Ziel-Lokation von *T*, gehe zu Schritt 1.

0. Initialisiere den HA:
   L := Anfangslokation
   P := Anfangspolyeder
1. Schneide *P*
   mit der Invariante von *L*.
2. Lasse *P* entsprechend der
   Aktivität von *L* wachsen.
3. Schneide *P*
   mit der Invariante von *L*.
4. Stop, falls *L* schon mit *P*
   besucht wurde. Falls nicht:

Für alle Transitionen*T* von *L*:

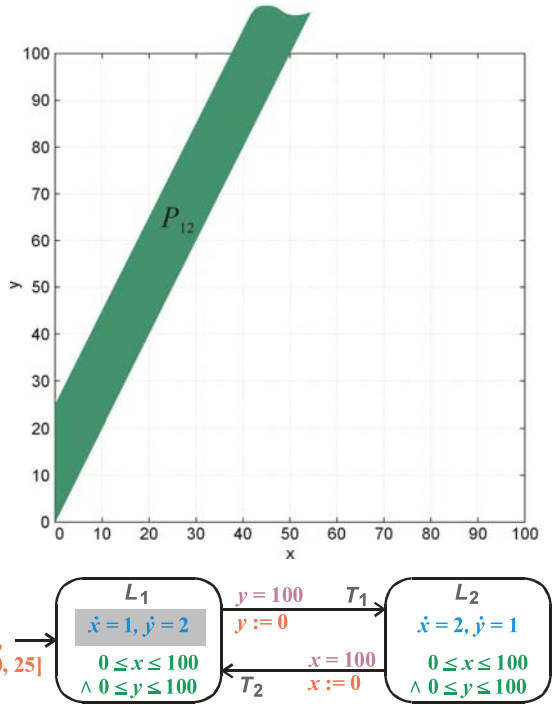
5. Schneide *P* mit dem
   Guard von *T*. Falls leer,
   nehme nächste Transition.
6. Falls nicht, setze *P* entspr.
   Reset-Ausdruck zurück.
7. Setze *L* := Ziel-Lokation
   von *T*, gehe zu Schritt 1.



0. Initialisiere den HA:
   L := Anfangslokation
   P := Anfangspolyeder
1. Schneide P
   mit der Invariante von L.
2. Lasse P entsprechend der
   Aktivität von L wachsen.
3. Schneide P
   mit der Invariante von L.
4. Stop, falls L schon mit P
   besucht wurde. Falls nicht:

Für alle Transitionen$T$ von L:

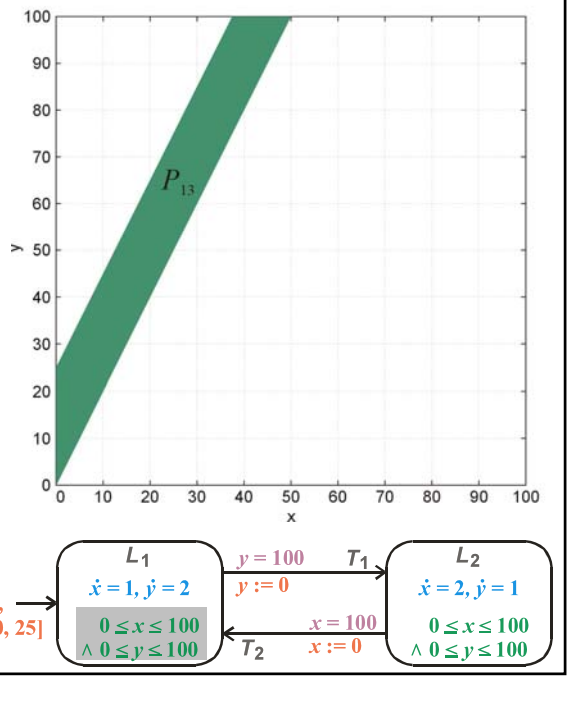
5. Schneide P mit dem
   Guard von T. Falls leer,
   nehme nächste Transition.
6. Falls nicht, setze P entspr.
   Reset-Ausdruck zurück.
7. Setze L := Ziel-Lokation
   von T, gehe zu Schritt 1.



0. Initialisiere den HA:
   L := Anfangslokation
   P := Anfangspolyeder
1. Schneide P
   mit der Invariante von L.
2. Lasse P entsprechend der
   Aktivität von L wachsen.
3. Schneide P
   mit der Invariante von L.
4. Stop, falls L schon mit P
   besucht wurde. Falls nicht:

Für alle Transitionen$T$ von L:

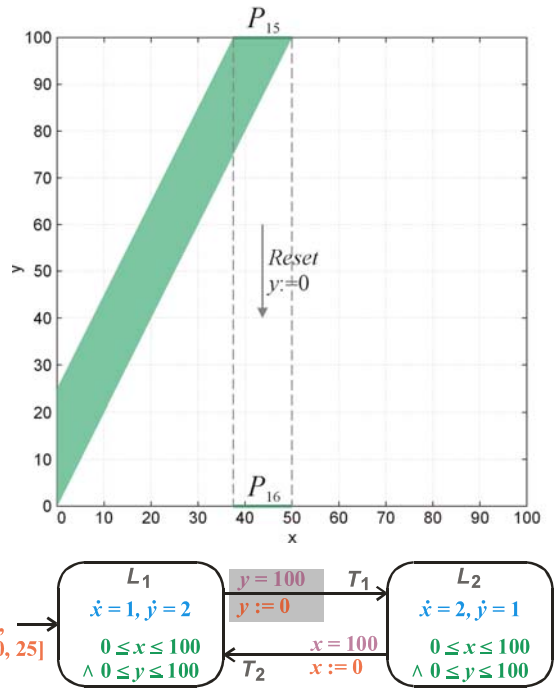
5. Schneide P mit dem
   Guard von T. Falls leer,
   nehme nächste Transition.
6. Falls nicht, setze P entspr.
   Reset-Ausdruck zurück.
7. Setze L := Ziel-Lokation
   von T, gehe zu Schritt 1.



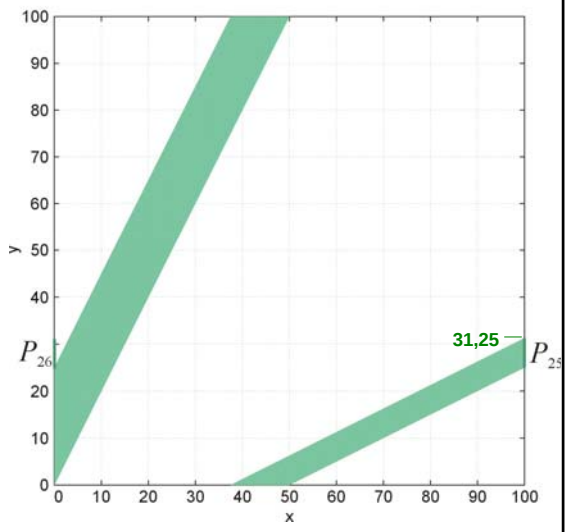
0. Initialisiere den HA:
   L := Anfangslokation
   P := Anfangspolyeder
1. Schneide *P*
   mit der Invariante von *L*.
2. Lasse *P* entsprechend der
   Aktivität von *L* wachsen.
3. Schneide *P*
   mit der Invariante von *L*.
4. Stop, falls *L* schon mit *P*
   besucht wurde. Falls nicht:

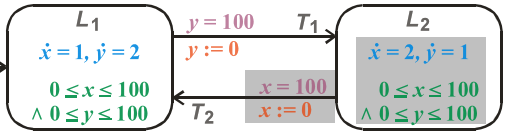
Für alle Transitionen*T* von *L*:

5. Schneide *P* mit dem
   Guard von *T*. Falls leer,
   nehme nächste Transition.
6. Falls nicht, setze *P* entspr.
   Reset-Ausdruck zurück.
7. Setze *L* := Ziel-Lokation
   von *T*, gehe zu Schritt 1.



$x := 0,$
$y := [0, 25]$





0. Initialisiere den HA:
   L := Anfangslokation
   P := Anfangspolyeder
1. Schneide *P*
   mit der Invariante von *L*.
2. Lasse *P* entsprechend der
   Aktivität von *L* wachsen.
3. Schneide *P*
   mit der Invariante von *L*.
4. Stop, falls *L* schon mit *P*
   besucht wurde. Falls nicht:

Für alle Transitionen*T* von *L*:

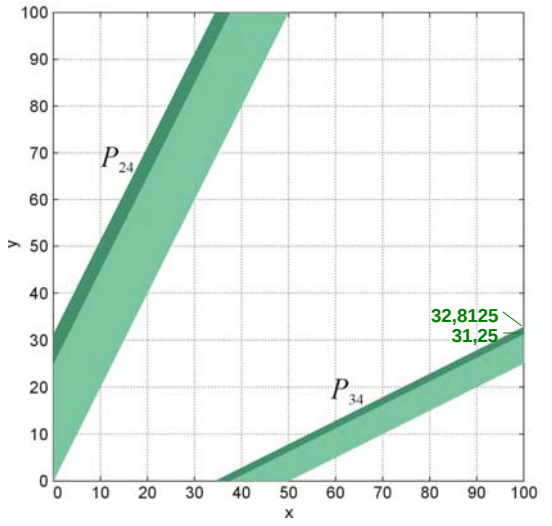
5. Schneide *P* mit dem
   Guard von *T*. Falls leer,
   nehme nächste Transition.
6. Falls nicht, setze *P* entspr.
   Reset-Ausdruck zurück.
7. Setze *L* := Ziel-Lokation
   von *T*, gehe zu Schritt 1.

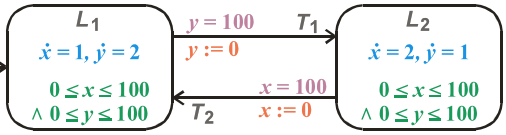
$x := 0,$
$y := [0, 25]$

0. Initialisiere den HA:
   L := Anfangslokation
   P := Anfangspolyeder
1. Schneide *P*
   mit der Invariante von *L*.
2. Lasse *P* entsprechend der
   Aktivität von *L* wachsen.
3. Schneide *P*
   mit der Invariante von *L*.
4. Stop, falls *L* schon mit *P*
   besucht wurde. Falls nicht:

Für alle Transitionen*T* von *L*:

5. Schneide *P* mit dem
   Guard von *T*. Falls leer,
   nehme nächste Transition.
6. Falls nicht, setze *P* entspr.
   Reset-Ausdruck zurück.
7. Setze *L* := Ziel-Lokation
   von *T*, gehe zu Schritt 1.

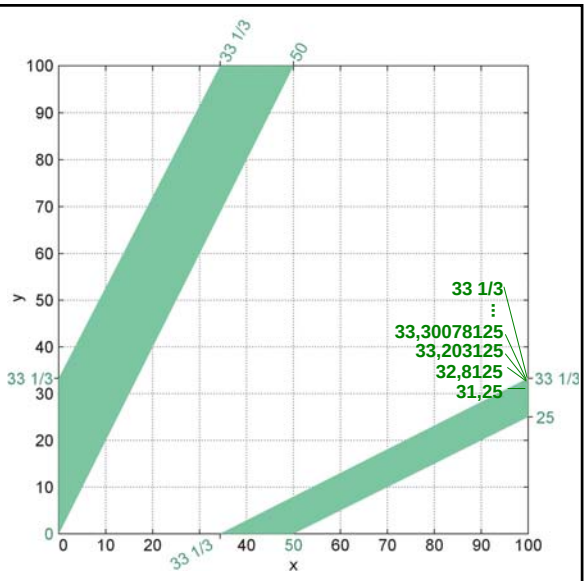


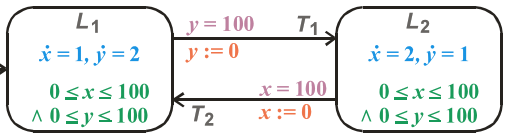


0. Initialisiere den HA:
   L := Anfangslokation
   P := Anfangspolyeder
1. Schneide *P*
   mit der Invariante von *L*.
2. Lasse *P* entsprechend der
   Aktivität von *L* wachsen.
3. Schneide *P*
   mit der Invariante von *L*.
4. Stop, falls *L* schon mit *P*
   besucht wurde. Falls nicht:

Für alle Transitionen*T* von *L*:

5. Schneide *P* mit dem
   Guard von *T*. Falls leer,
   nehme nächste Transition.
6. Falls nicht, setze *P* entspr.
   Reset-Ausdruck zurück.
7. Setze *L* := Ziel-Lokation
   von *T*, gehe zu Schritt 1.

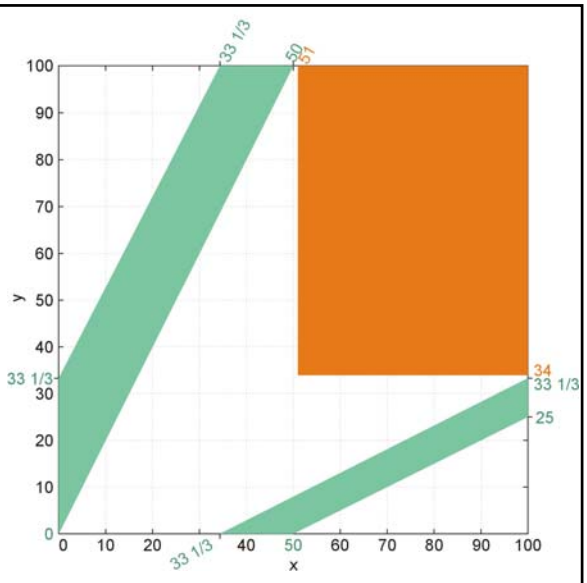


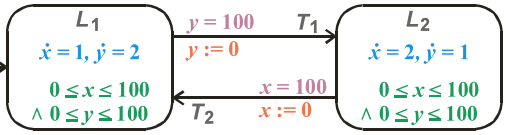


0. Initialisiere den HA:
   L := Anfangslokation
   P := Anfangspolyeder
1. Schneide *P*
   mit der Invariante von *L*.
2. Lasse *P* entsprechend der
   Aktivität von *L* wachsen.
3. Schneide *P*
   mit der Invariante von *L*.
4. Stop, falls *L* schon mit *P*
   besucht wurde. Falls nicht:

Für alle Transitionen *T* von *L*:

5. Schneide *P* mit dem
   Guard von *T*. Falls leer,
   nehme nächste Transition.
6. Falls nicht, setze *P* entspr.
   Reset-Ausdruck zurück.
7. Setze *L* := Ziel-Lokation
   von *T*, gehe zu Schritt 1.





Erreichbarkeitsanalyse von HA: Grenzen

Theoretisch:

- Erreichbarkeit ist unentscheidbar für allgemeine HA.
- Entscheidbarkeit gilt nur für sehr eingeschränkte Klassen von HA (Alur et al., 1995), z.B. Echtzeitautomaten.
- Wo ist die Grenze?

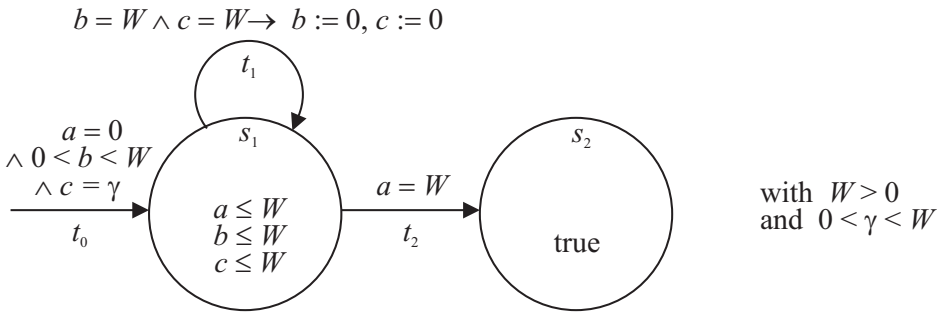
→ Zufügen nur einer Stoppuhr zu einem Echtzeitautomaten macht Erreichbarkeit unentscheidbar.

Praktisch:

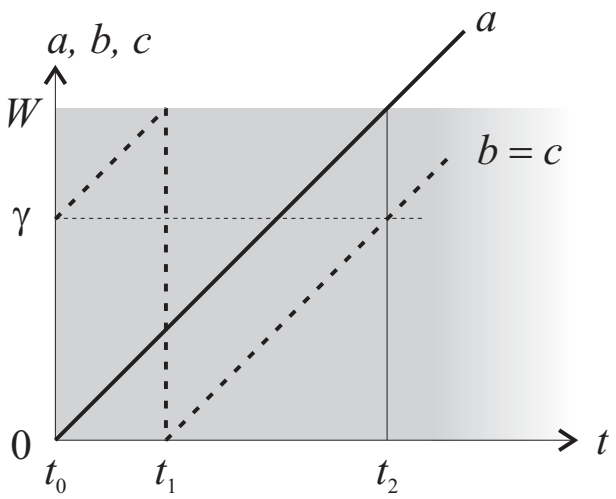
- Hytech verwendet ganzzahlige Arithmetik (kein Runden).
  ⇒ Integer-Überläufe nach wenigen Iterationen.
- Rechenzeit wächst exponentiell mit der Anzahl kontinuierlicher Variablen (zusätzlich zur diskreten Zustandsraumexplosion).

Abhilfe: **Abstraktion**

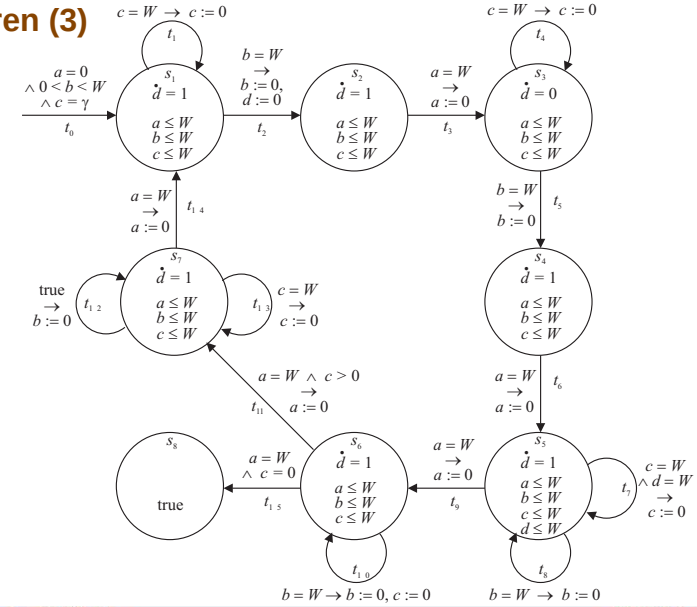
Beispiel für Nichttermination der Erreichbarkeit wegen Stoppuhren (1)



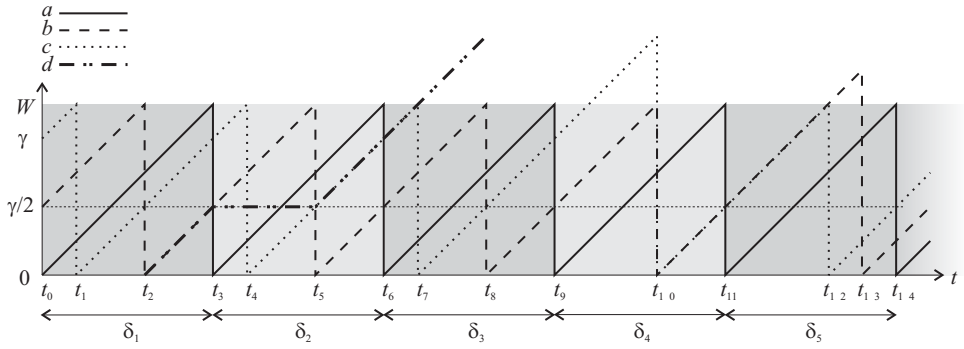
Beispiel für Nichttermination der Erreichbarkeit wegen Stoppuhren (2)



Beispiel für Nichttermination der Erreichbarkeit wegen Stoppuhren (3)

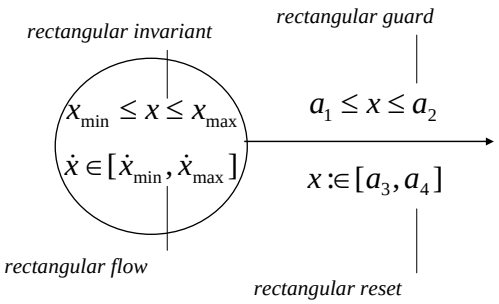


Beispiel für Nichttermination der Erreichbarkeit wegen Stoppuhren (4)



- δ_1: Synchronizes b and d.
- δ_2: $c(t_0) = 2b(t_0) \Rightarrow d$ synchronized with c.
- δ_3: Ensures that $c(t_6) = d(t_6)$, which requires $c(t_3) = 2b(t_3)$.
- δ_4: Synchronizes b and $d \Rightarrow c(t_{11}) = b(t_9)$. From t_0 to t_{11}: c:=c/2
- δ_5: Choose new value for b. (Must be $b(t_{14}) = b(t_{11})/2$ to avoid deadlock)

Rektangulärer Automat

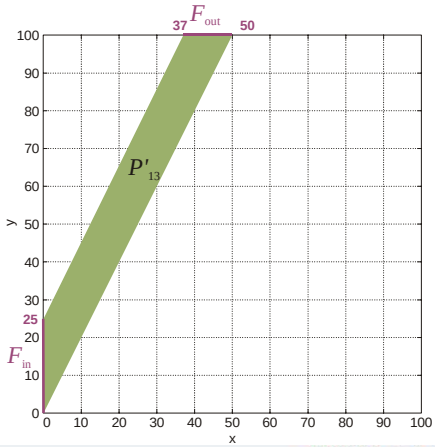


Grundidee bei der approximativen Analyse einfacher rektangulärer Automaten

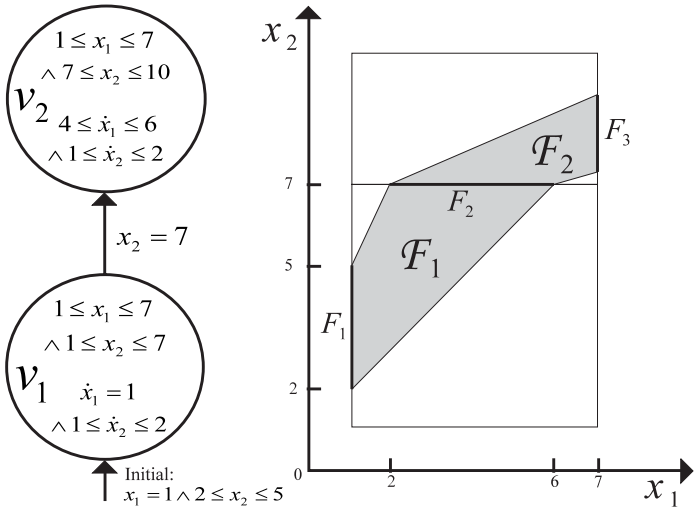
- Implizite Darstellung der erreichbaren Regionen R durch ihre Eingangs- und Ausgangs-Faces F_{in} bzw. F_{out}. Es gilt: R = ***KonvexeHülle***(F_{in}, F_{out})

Berechnungsvorschrift (Beispiel):

1. Bestimme das Zeitintervall ΔT, in dem die Grenze y = **100** von Punkten aus F_{in} erreicht werden kann:
   $\dot{y} = 2 \Rightarrow \Delta T = [37.5, 50]$
2. Runde konservativ auf gewünschte Auflösung (= „Dehnen“ von ΔT):
   $\Delta T' = [37, 50]$
3. Berechne für die anderen Dimensionen die in ΔT mögliche Auslenkung:
   $\dot{x} = 1 \Rightarrow F_{out,\,x} = [37, 50]$



Approximative Erreichbarkeitsanalyse mit „Faces“

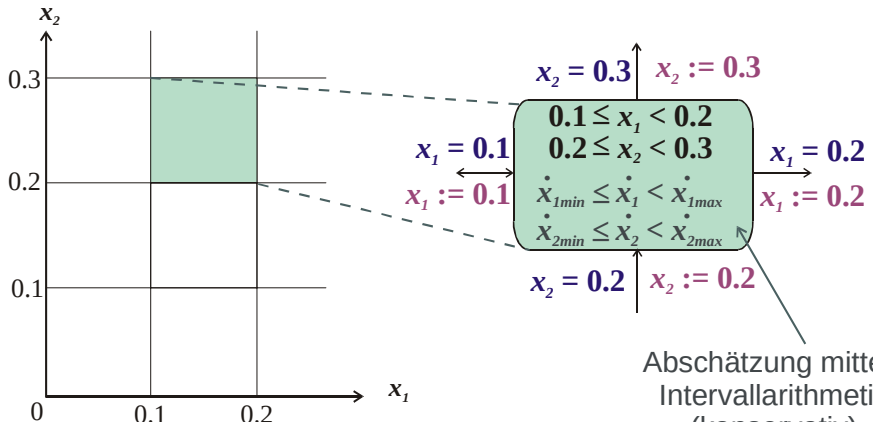


Abstraktion von geschalteten kontinuierlichen Systemen durch rektanguläre Automaten

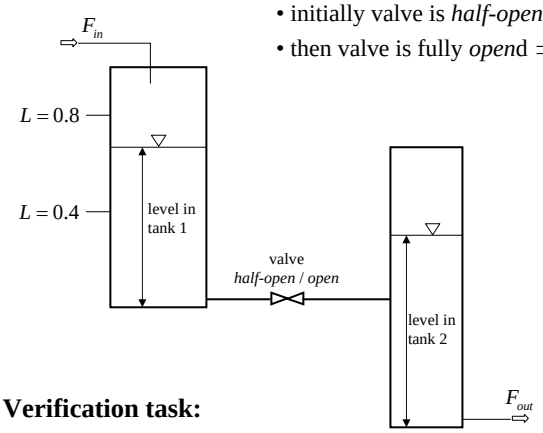
Geschaltetes kontinuierliches System:

$\dot{\underline{x}} = f(\underline{x}, u)$

Rektangulärer Automat:



Two-Tank Example

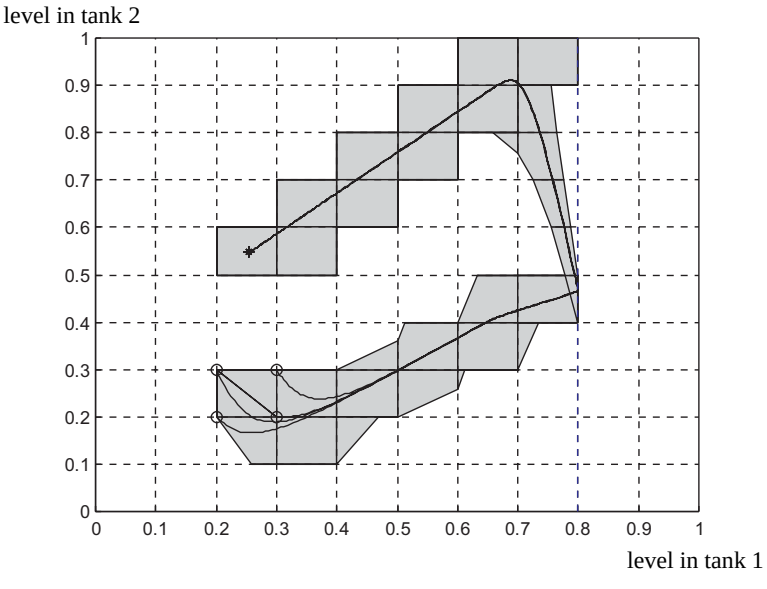


- constant inlet flow F_{in}
- initially valve is *half-open* $\Rightarrow$ level in tank 1 rises
- then valve is fully *open*d $\Rightarrow$ level in tank 1 falls and level in tank 2 may rise until equilibrium $F_{in} = F_{out}$ is reached

Verification task:

When a certain **level *L*** in tank 1 is **reached**, the **controller switches** the valve to *open* to avoid an overflow in tank 1.
Can this lead to an **overflow in tank 2?**

Analysis Results for $L = 0.8$



(Unexpected) Analysis Results for $L = 0.4$