Formale Methoden für eingebettete Systeme
Teil 9: Hybride Systeme

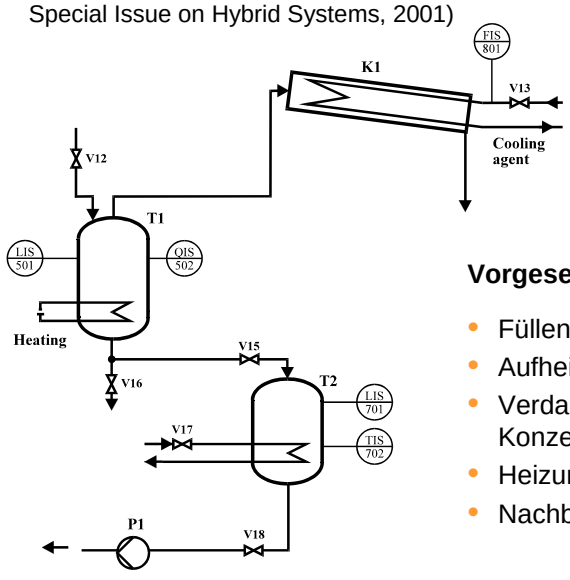
Prof. Dr. Stefan Kowalewski

Lehrstuhl Informatik 11
RWTH Aachen

Sommersemester 2005

Beispiel

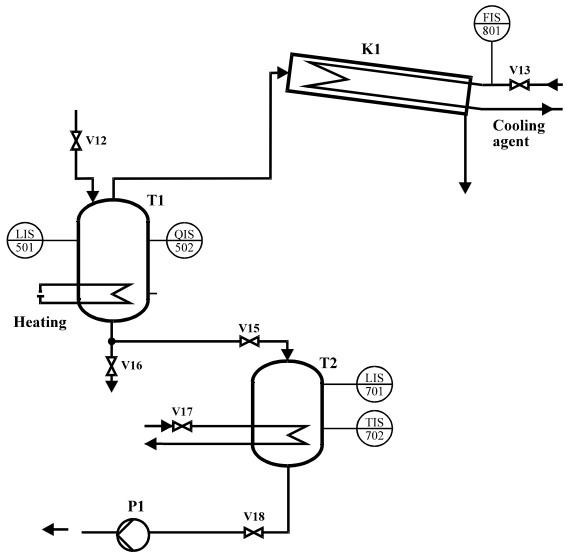
(Kowalewski et al., Europ. Journal of Control, Special Issue on Hybrid Systems, 2001)



Vorgesehener Produktionsablauf:

- Füllen von Behälter T1
- Aufheizen
- Verdampfen bis gewünschte Konzentration erreicht ist
- Heizung ausschalten und T1 leeren
- Nachbearbeitungsschritt in T2

Notabschaltung bei Kühlungsausfall



Randbedingungen:

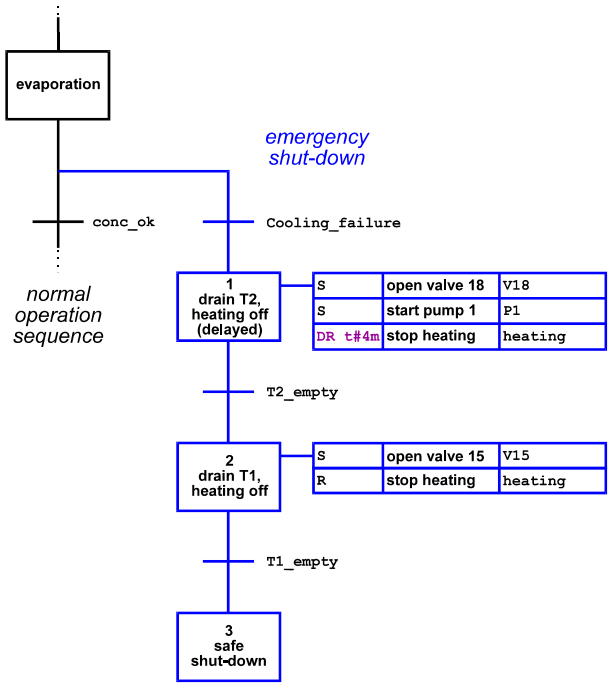
1. Bei fortgesetzter Wärmezufuhr steigen Temperatur und Druck (geschlossenes System).
2. Bei Abkühlen unter einen Grenzwert beginnt das Material im Verdampfer zu kristallisieren.

→ **Gegenläufige Anforderungen:**

- Heizung muss **früh genug** abgeschaltet werden, um gefährlichen Druckanstieg zu vermeiden.
- Heizung muss **lange genug** eingeschaltet bleiben, um Kristallisierung zu vermeiden.

Vorschlag für Steuerungs-programm

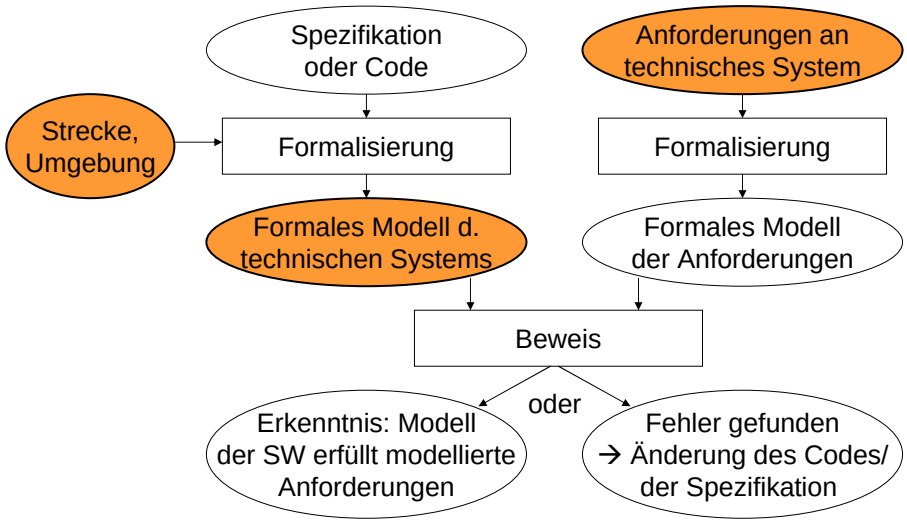
Sequential Function Chart (SFC) nach IEC 1131-3



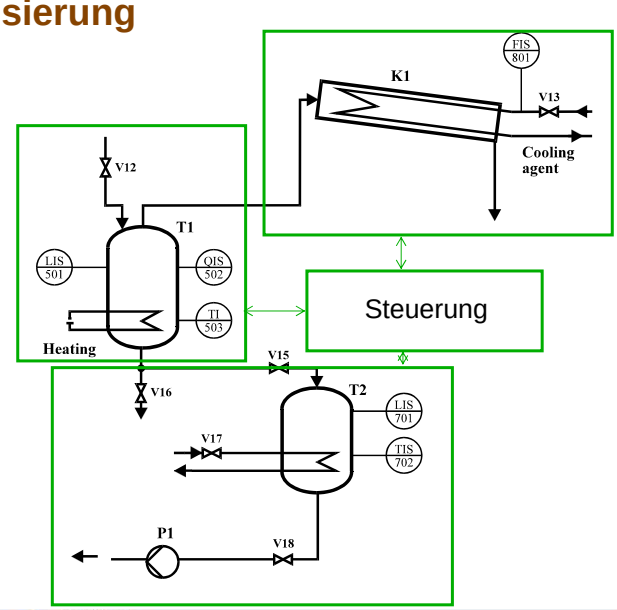
Problemstellung

- Erfüllt das Steuerungsprogramm die folgenden Anforderungen?
  - Der Druck darf oberen Grenzwert nicht überschreiten.
  - Die Temperatur darf Kristallisationspunkt nicht unterschreiten.
- Zu überprüfen ist:
  - logischer Entwurf des SFCs
  - Wahl des Wertes für den Parameter Wartezeit (t#4m)
- Anforderungen sind **für den gesteuerten Prozess** ("geschlossenen Kreis", "technisches System") und nicht für das Steuerungsprogramm formuliert.

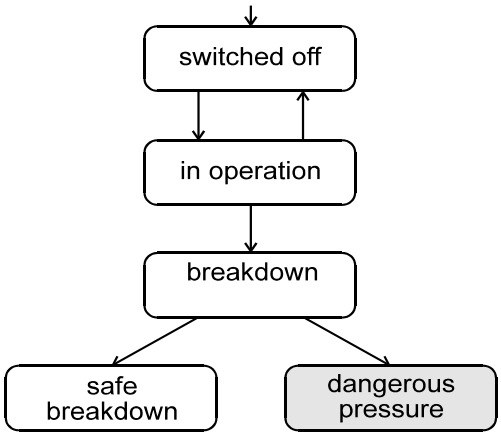
Formale Verifikation von Automatisierungssystemen



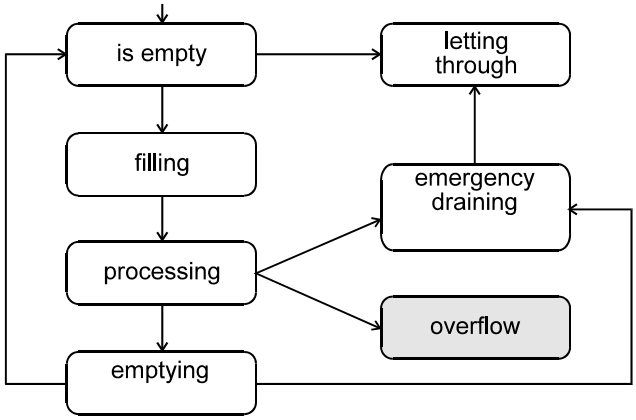
Formalisierung



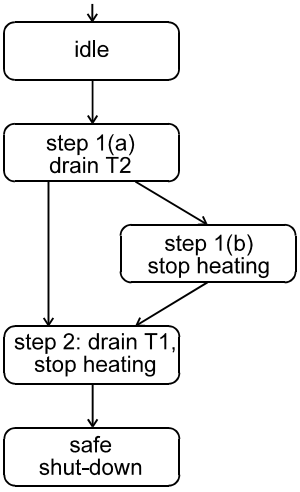
Diskretes Modell des Kondensators K1



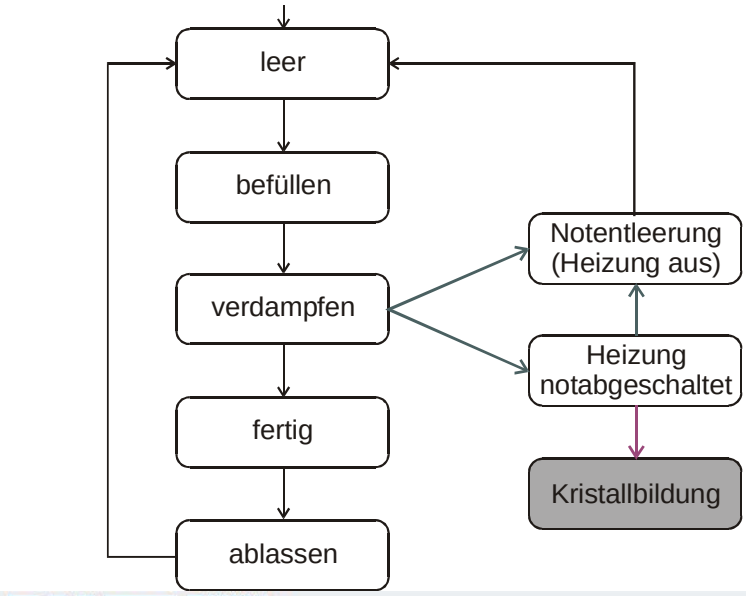
Diskretes Modell von Behälter T2



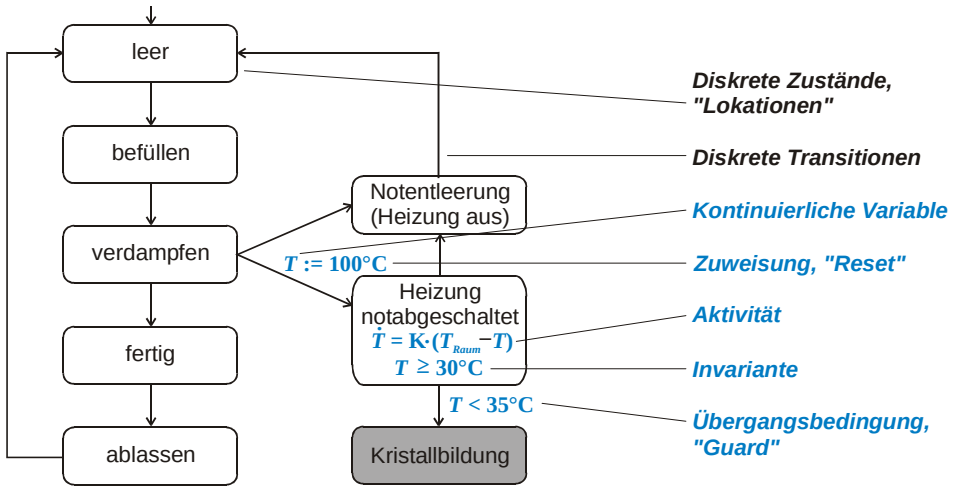
Diskretes Modell der Steuerung



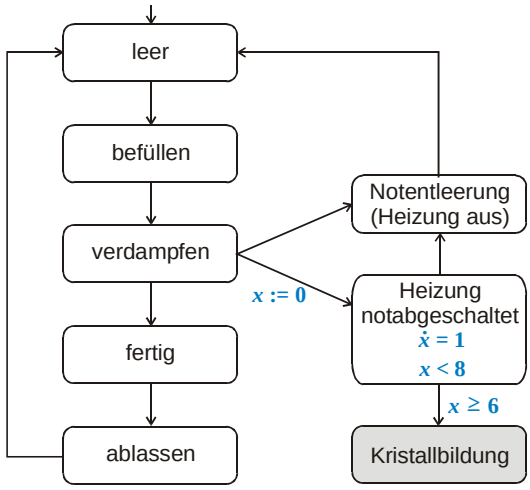
Diskretes Modell für Tank 1



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.

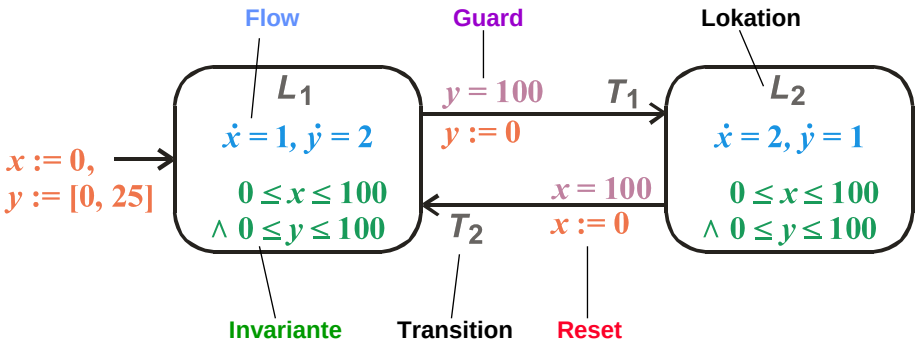
Frage:

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

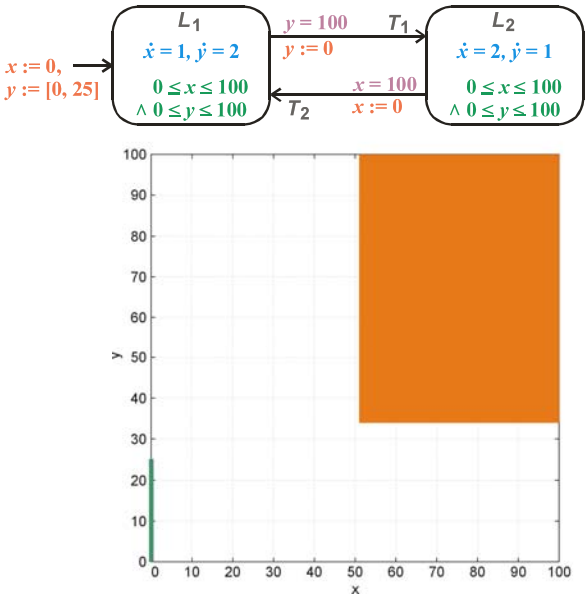
Werkzeuge:

- ***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:

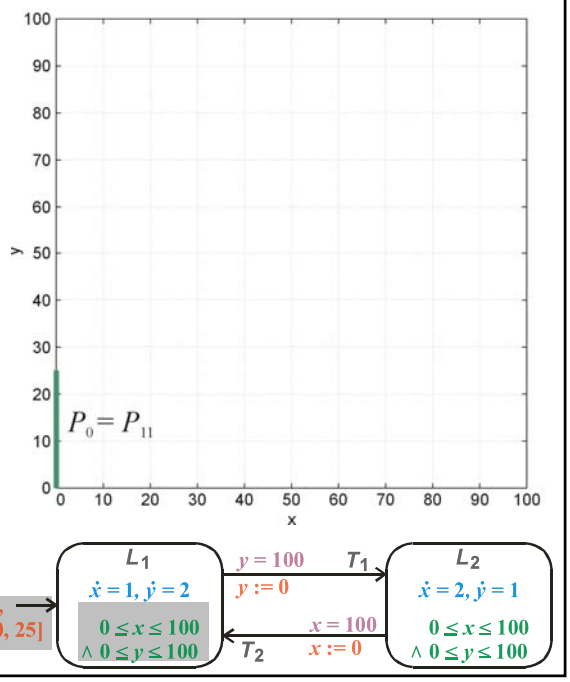
Für alle Transitionen T von L:

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:

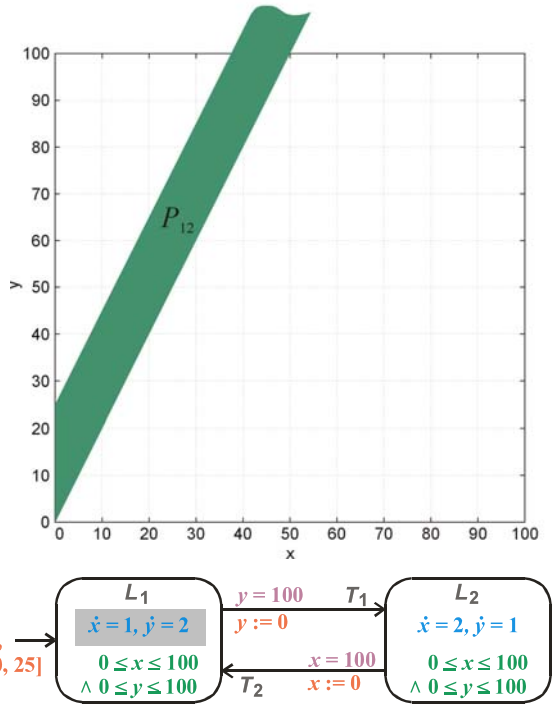
5. Schneide P mit dem
Guard von T. Falls leer,
nehme nächste Transition.
6. Falls nicht, setze P entspr.
Reset-Ausdruck zurück.
7. Setze L := Ziel-Lokation
von T, gehe zu Schritt 1.



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:

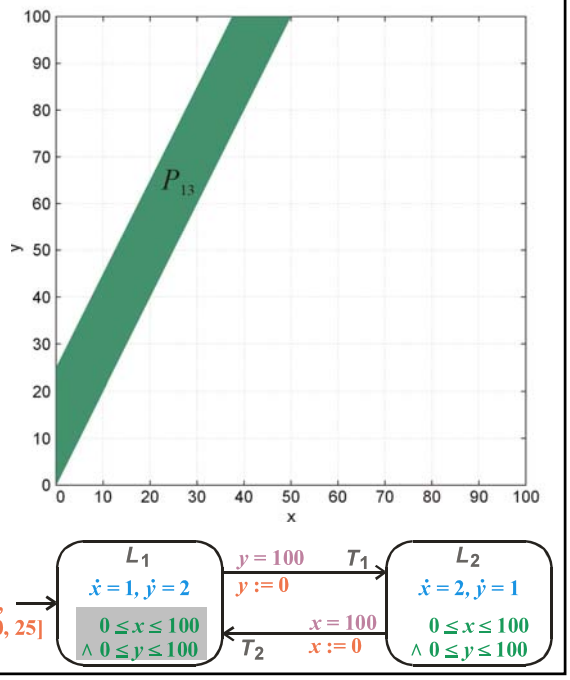
5. Schneide P mit dem
   Guard von T. Falls leer,
   nehme nächste Transition.
6. Falls nicht, setze P entspr.
   Reset-Ausdruck zurück.
7. Setze L := Ziel-Lokation
   von T, gehe zu Schritt 1.



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:

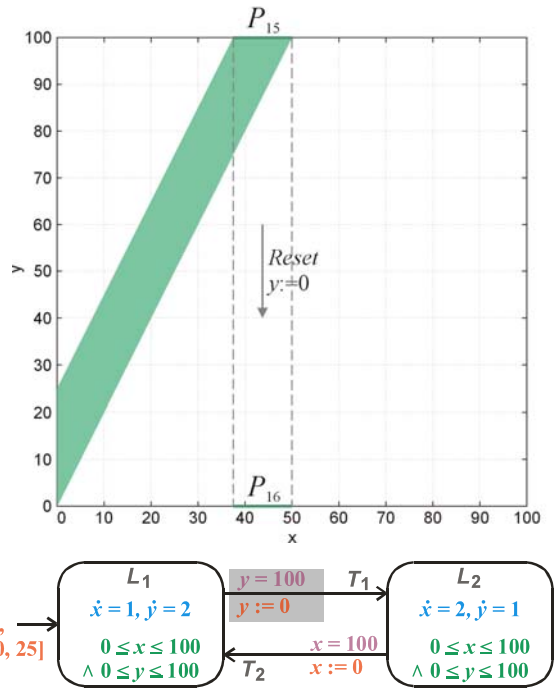
5. Schneide P mit dem
   Guard von T. Falls leer,
   nehme nächste Transition.
6. Falls nicht, setze P entspr.
   Reset-Ausdruck zurück.
7. Setze L := Ziel-Lokation
   von T, gehe zu Schritt 1.



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.

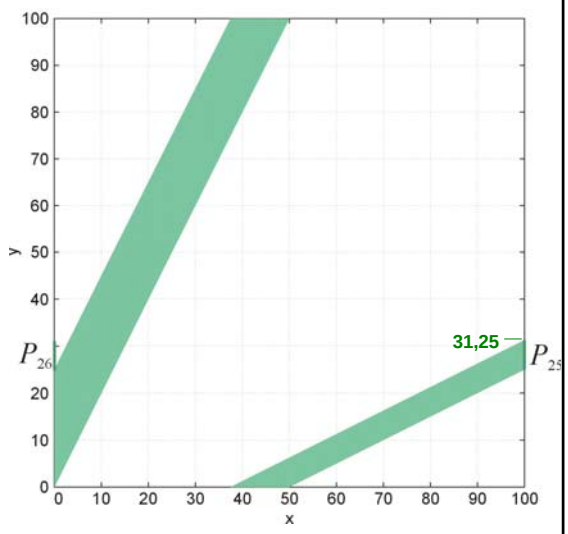


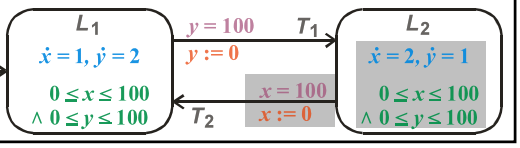
$x := 0,$
$y := [0, 25]$

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.



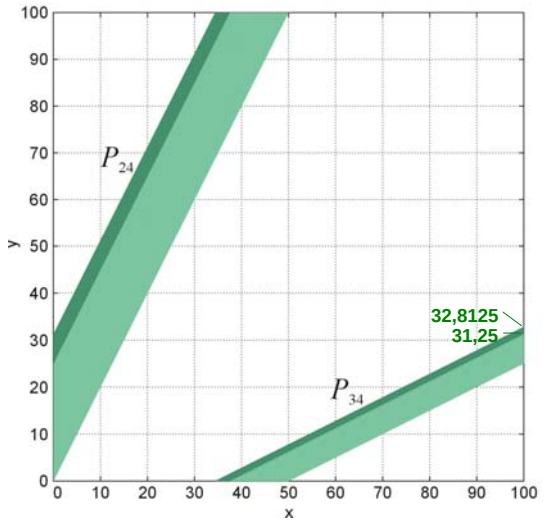


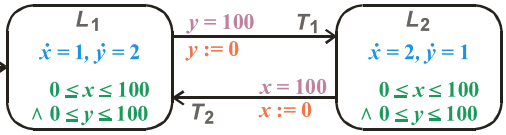
$x := 0,$
$y := [0, 25]$

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.

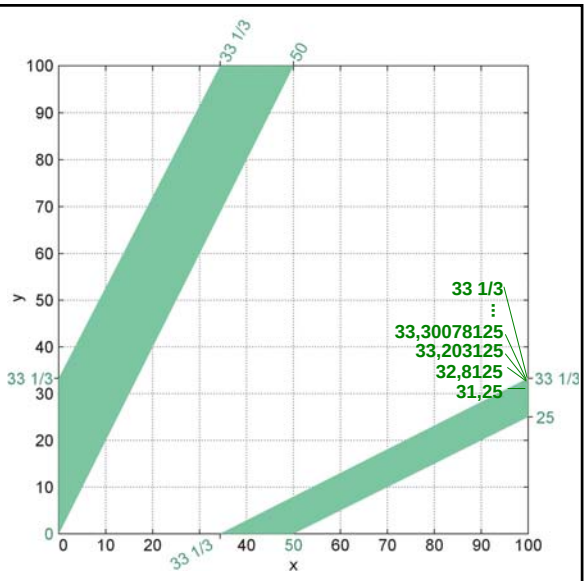


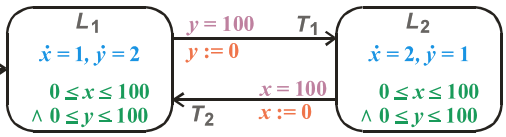


0. Initialisiere den HA:
   L := Anfangslokation
   P := Anfangspolyeder
1. Schneide *P*
   mit der Invariante von *L*.
2. Lasse *P* entsprechend der
   Aktivität von *L* wachsen.
3. Schneide *P*
   mit der Invariante von *L*.
4. Stop, falls *L* schon mit *P*
   besucht wurde. Falls nicht:

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.

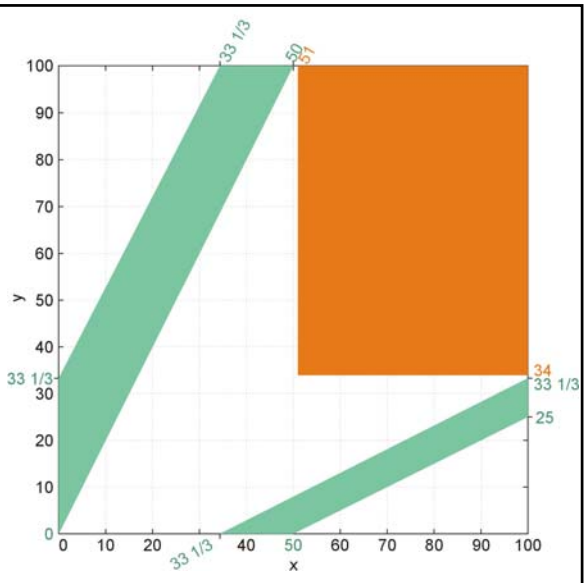


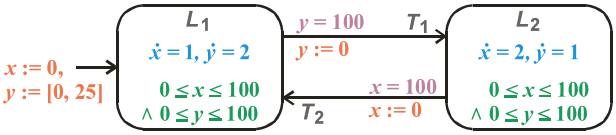


0. Initialisiere den HA:
   L := Anfangslokation
   P := Anfangspolyeder
1. Schneide P
   mit der Invariante von L.
2. Lasse P entsprechend der
   Aktivität von L wachsen.
3. Schneide P
   mit der Invariante von L.
4. Stop, falls L schon mit P
   besucht wurde. Falls nicht:

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).
- Für rektanguläre Automaten ist die Erreichbarkeit partiell entscheidbar.

Praktisch:

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

Abhilfe: **Abstraktion**