Ludwig-Maximilians-Universität München

Lehr- und Forschungseinheit Theoretische Informatik
Hauptseminar Programmanalyse
SS 2009

Abstrakte Interpretation I

Prof. Dr. Martin Hofmann
Referent: N.Mikhaylova

Gliederung

Prof. Dr. Patrick Cousot, Informatikprofessor an der École Normale Supérieure in Paris

Abstrakte Interpretation - ist eine allgemeine Theorie für semantische Approximation von diskreten dynamischen Systemen, z.B Berechnung eines Programms

Abstrakte Interpretation in der Programmanalyse:

- Model-Checking
- Approximation der Fixpunkte
- Software Steganographie
- Statische Analyse

Programme können häufig nicht vollständig analysiert werden:

- Es gibt unendlich viele Eingabewerte
- Variablen können die Belegungen aus einem unendlich großem Werteraum haben

Grundidee der abstrakten Interpretation:

das Programm nicht auf den eigentlichen „konkreten“ Werten, sondern auf Abstraktionen der Datentypen analysieren

Abstraktion in Galois Verbindungen

Es werden zwei Funktionen definiert:

- Abstraktion α, die jeden Wert auf auf seinen abstrakten Wert abbildet

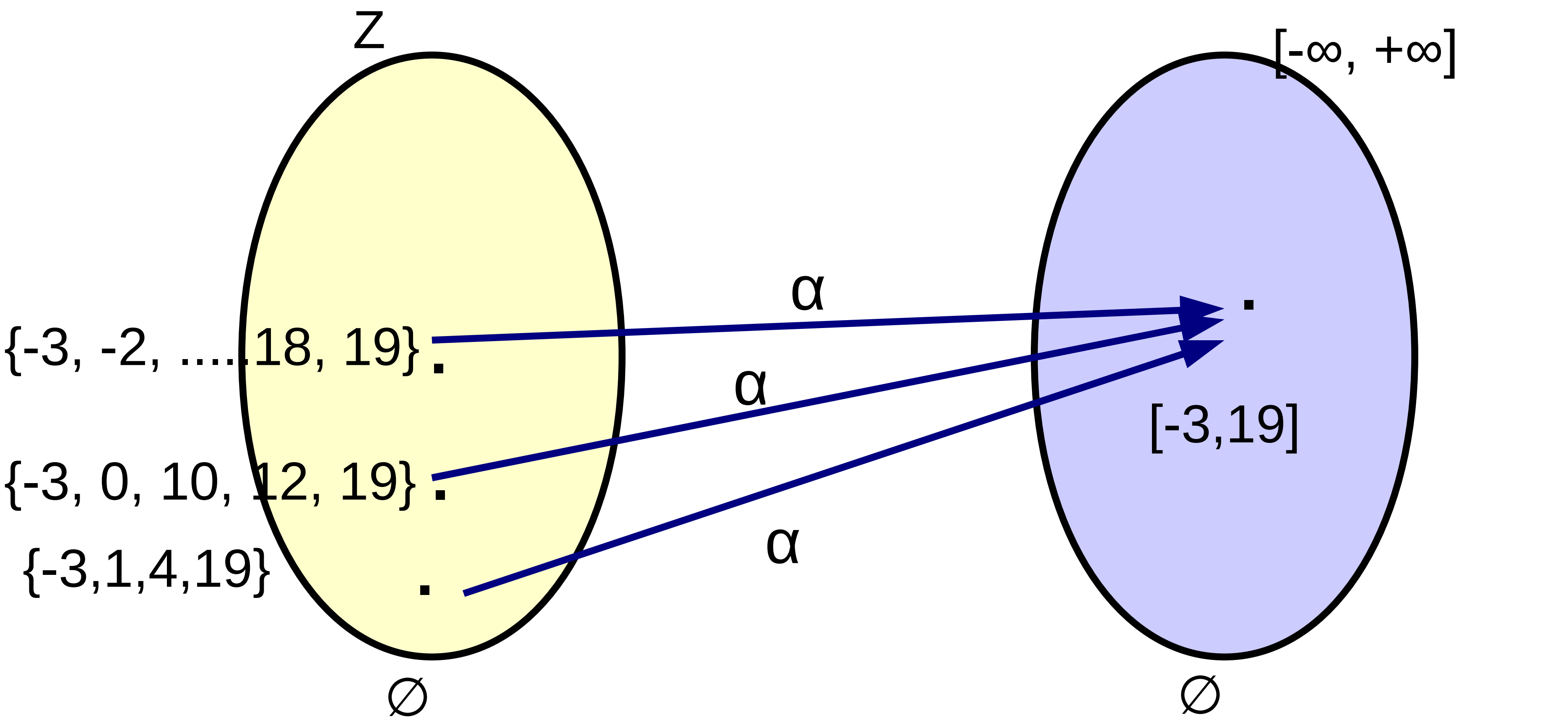
$$\alpha : L \rightarrow M$$

- Konkretisierung γ, die einem abstrakten Wert alle konkrete Werte zuordnet, für die er steht

$$\gamma : M \rightarrow L$$

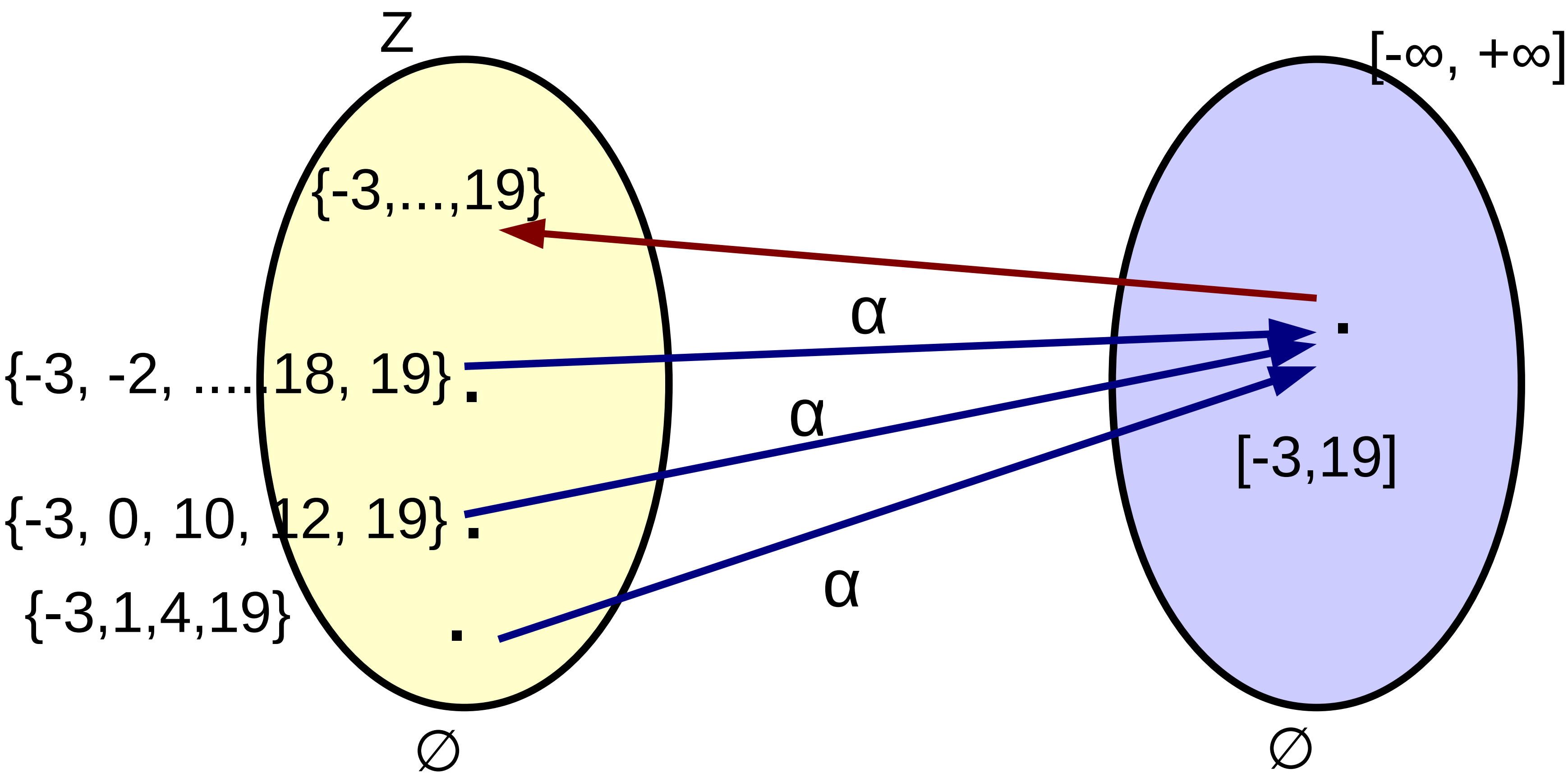
Abstraktion in Galois Verbindungen

konkreter Bereich abstrakter Bereich



Abstraktion in Galois Verbindungen

konkreter Bereich   abstrakter Bereich



Möglichkeiten der Abstraktion

- Zahlen durch even/odd abstrahieren

  α: Z → {even, odd}

$$\alpha(x) = \begin{cases} \{\text{even}\}, & \text{falls x gerade} \\ \{\text{odd}\}, & \text{falls x ungerade} \end{cases}$$

(B. König)

Beispiel „Korrektheit der arithmetischen Berechnung“:

$$373 * 8847 + 12345 = 3312266$$

Beispiel „Korrektheit der arithmetischen Berechnung“:

$$373 \ast 8847 + 12345 = 3312266$$

- eine Zahl ist durch 9 teilbar, genau dann wenn ihre Quersumme durch 9 teilbar ist

  $[a+b \bmod 9] = [((a \bmod 9)+(b \bmod 9)) \bmod 9]$

$$373 \ast 8847 + 12345 = 3312266$$

$L = \mathbb{N}$

$M = \{0, \ldots, 8\}$

$\alpha(x) = x \bmod 9$

$\alpha_1 \oplus \alpha_2 = (\alpha_1 + \alpha_2) \bmod 9$

$\alpha_1 \circledast \alpha_2 = (\alpha_1 * \alpha_2) \bmod 9$

$l_1 * l_2 + l_3 = l_4 \Rightarrow \alpha(l_1) \circledast \alpha(l_2) \oplus \alpha(l_3) = \alpha(l_4)$

$$373 \ast 8847 + 12345 = 3312266$$

- QS[373] 13 mod 9 = 4
- QS[8847] 27 mod 9 = 0
- QS[12345] 15 mod 9 = 6
- QS[3312266] 23 mod 9 = 5

$$373 \ast 8847 + 12345 = 3312266$$

- QS[373] 13 mod 9 = 4
- QS[8847] 27 mod 9 = 0
- QS[12345] 15 mod 9 = 6
- QS[3312266] 23 mod 9 = 5

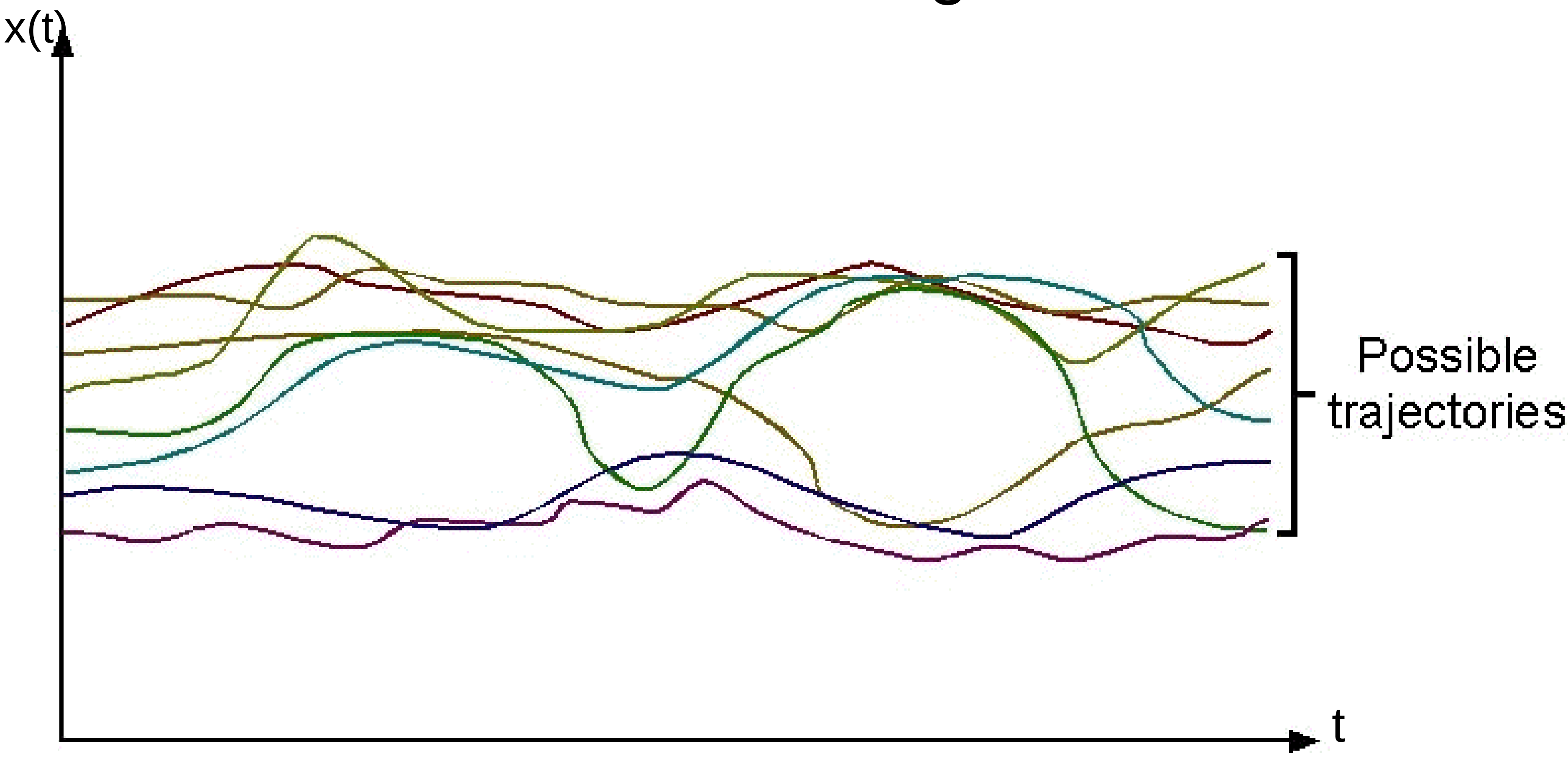
  $4 * 0 + 6 = 5$

  $6 \neq 5$

Die Anwendung der abstrakten Interpretation intuitiv:

Durch die abstrakte Semantik wird schließlich ein Gleichungssystem aufgebaut, dass für jeden Programmpunkt alle mögliche Zustände des Programms in allen möglichen Umgebungen berechnet

Konkrete Semantik des Programms



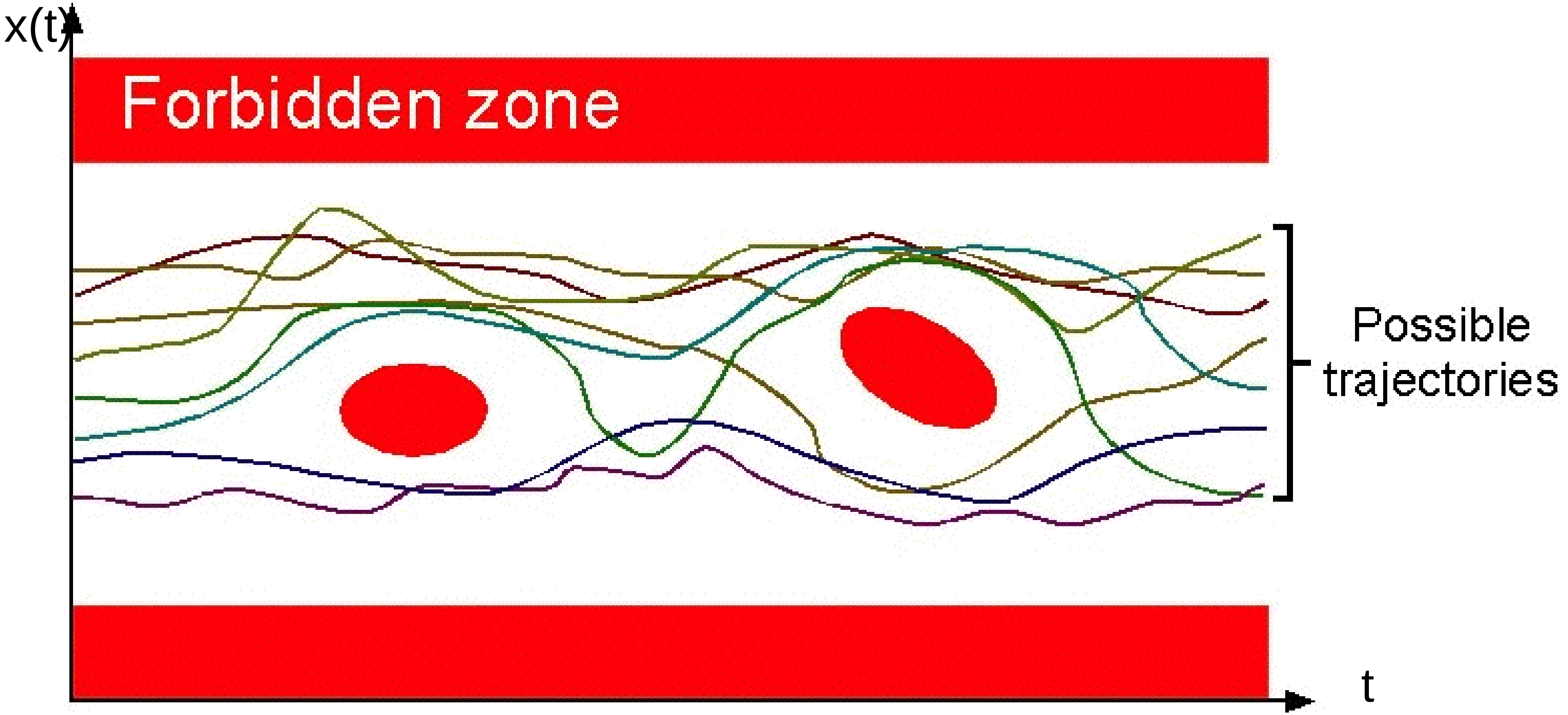
Unentscheidbarkeit

- konkrete Semantik des Programms ist im Allgemeinen als ein unendliches mathematisches Objekt nicht berechenbar
- Viele Fragen bezüglich konkreter Semantik des Programms sind unentscheidbar (z.B Terminierung des Programms)

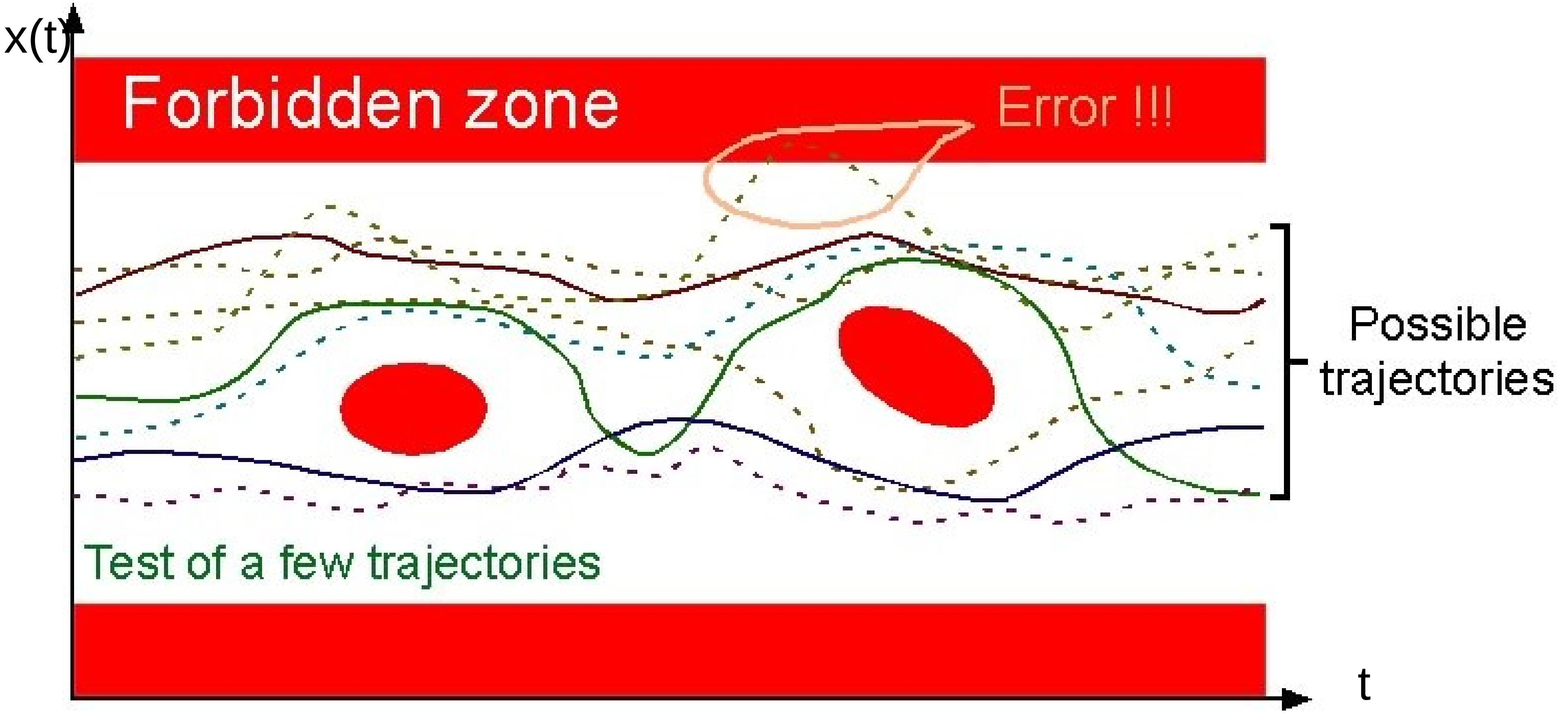
Prüfen die sicheren Pfaden

- die Verifikation der sicheren Pfaden besteht in Überprüfung, ob die konkrete Semantik sich mit der unzulässigen Zonen nicht überkreuzt

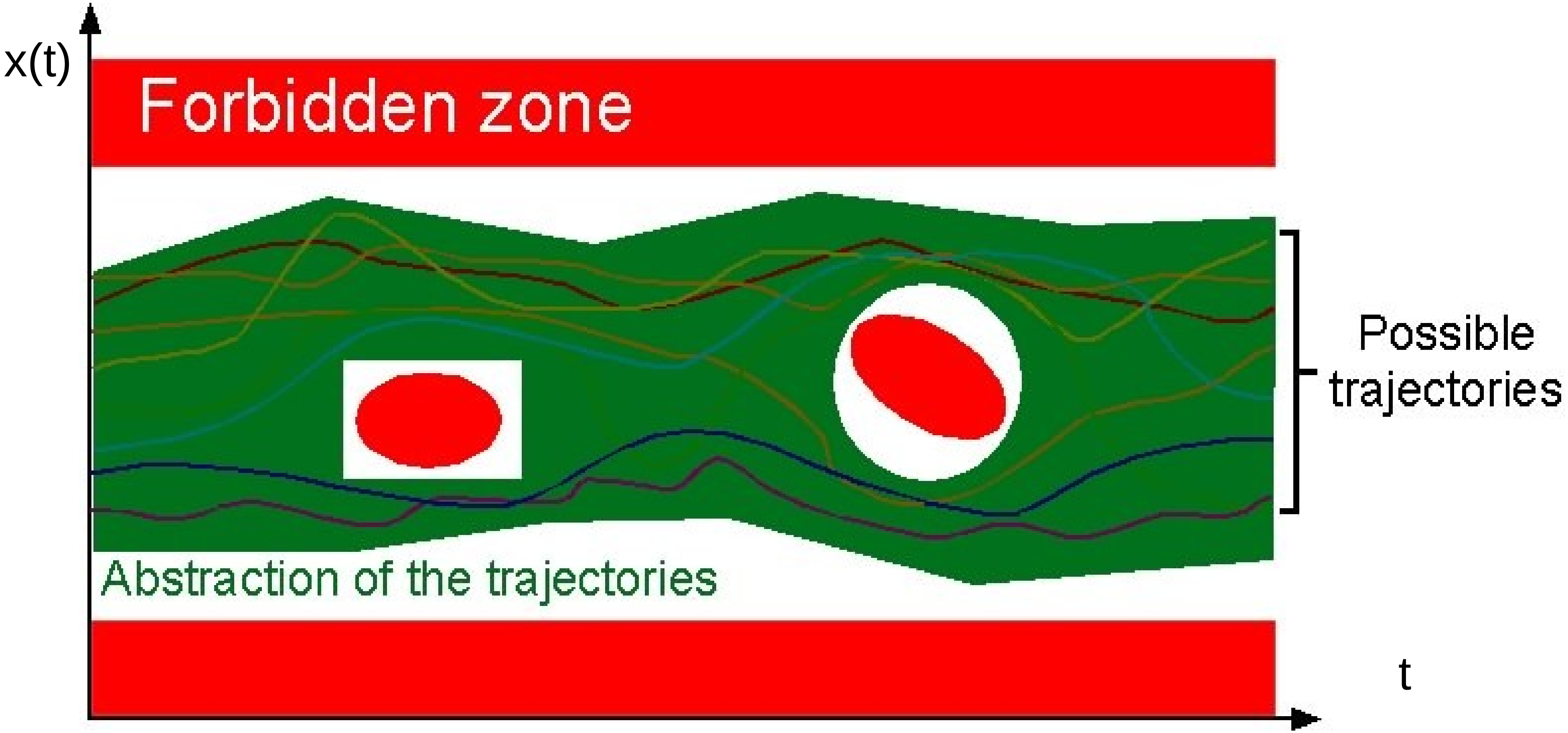
Sicherheitseingenschaften



Testen


x(t)
Forbidden zone
Error !!!
Possible trajectories
Test of a few trajectories
t

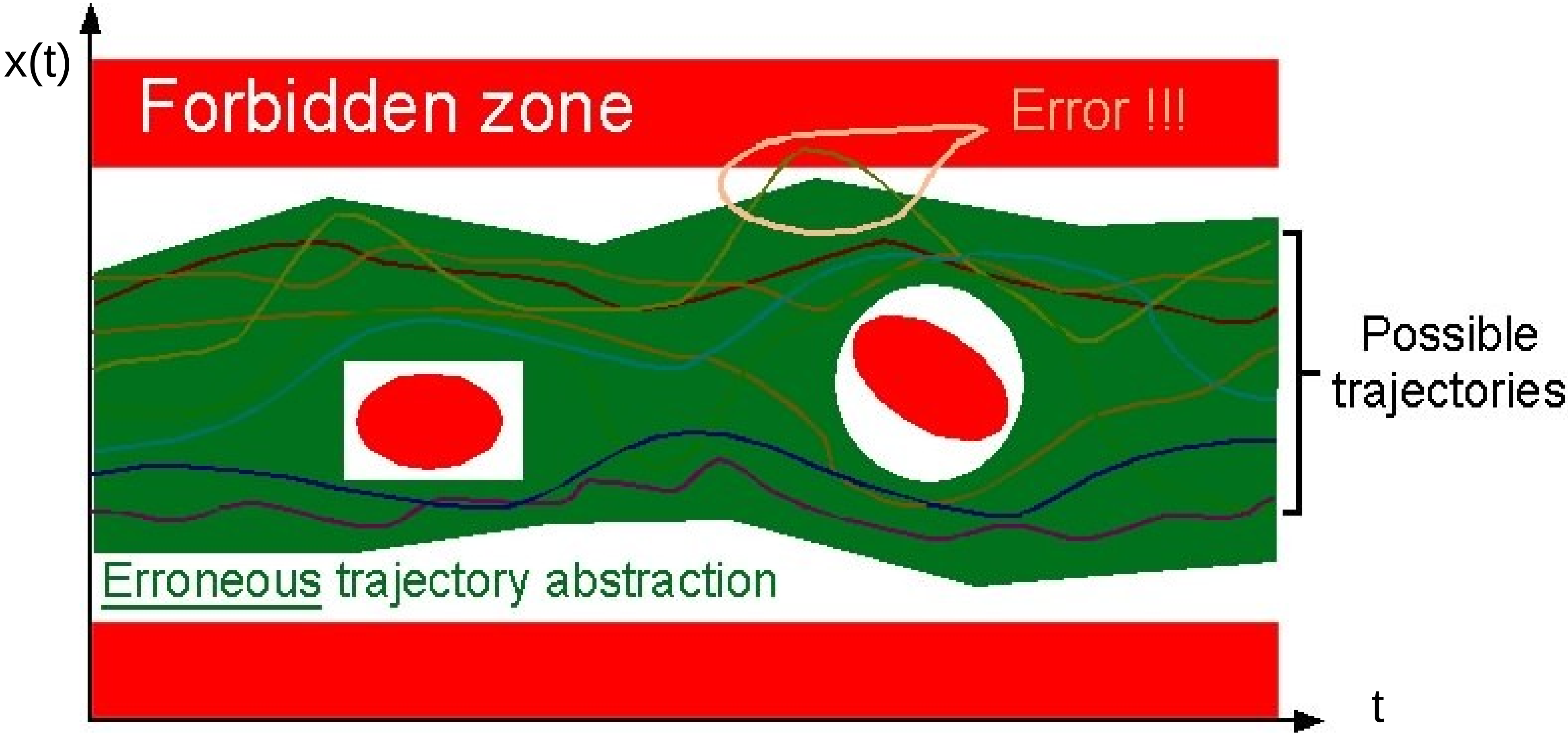
Abstrakte Interpretation



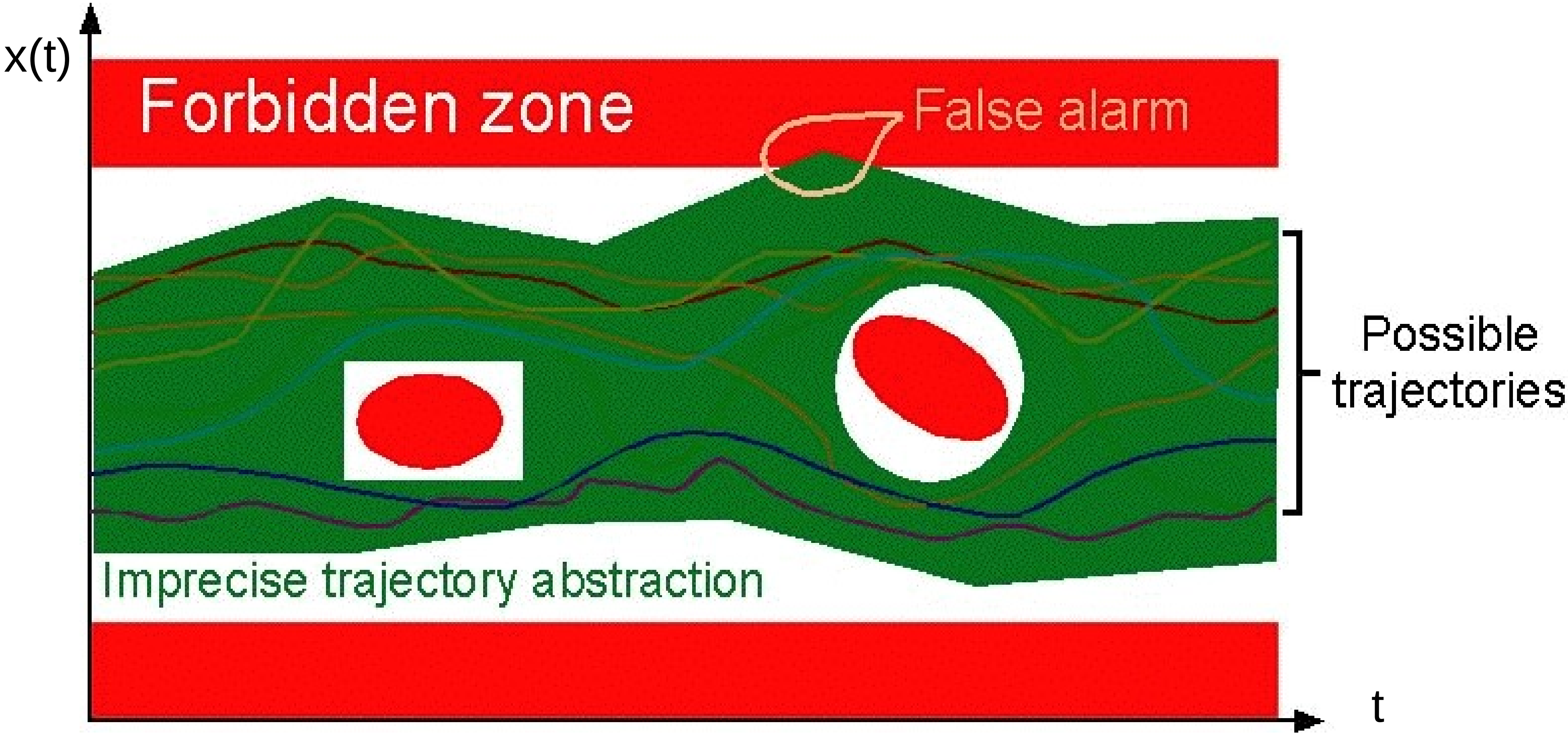
Formale Methoden

- Model-Checking - Technik zum Beweisen der Gültigkeit einer Spezifikationen im Bezug auf ein Programm
- Deduktive Methoden – spezifizieren die Verifikation der Bedingungen mit induktiven Schritten (Verwendung der Logikkalküls und Beweisregeln)
- Statische Analyse –abstrakte Semantik wird automatisch errechnet durch die vordefinierte Approximation

Fehlerhafte Abstraktion



Falscher Alarm



Abstrakte Interpretation in der Programmanalyse:

- Model-Checking
- Approximation der Fixpunkte
- Software Steganographie
- Statische Analyse

II. Approximation der Fixpunkte

II. Approximation der Fixpunkte

Begriffe:

- Vollständiger Verband (complete lattices)
- Fixpunkte
- Upper Bound Operator
- Widening
- Narrowing

Vollständiger Verband

Ein Tupel (L,$\sqsubseteq$), bestehend aus einer Menge L und einer partiellen Ordnung auf L heißt vollständiger Verband, wenn jede Teilmenge Y von L eine kleinste obere (Supremum) und eine größte untere (Infimum) Schranke hat. Dies muss insbesondere für Y = $\emptyset$ gelten. Man definiert

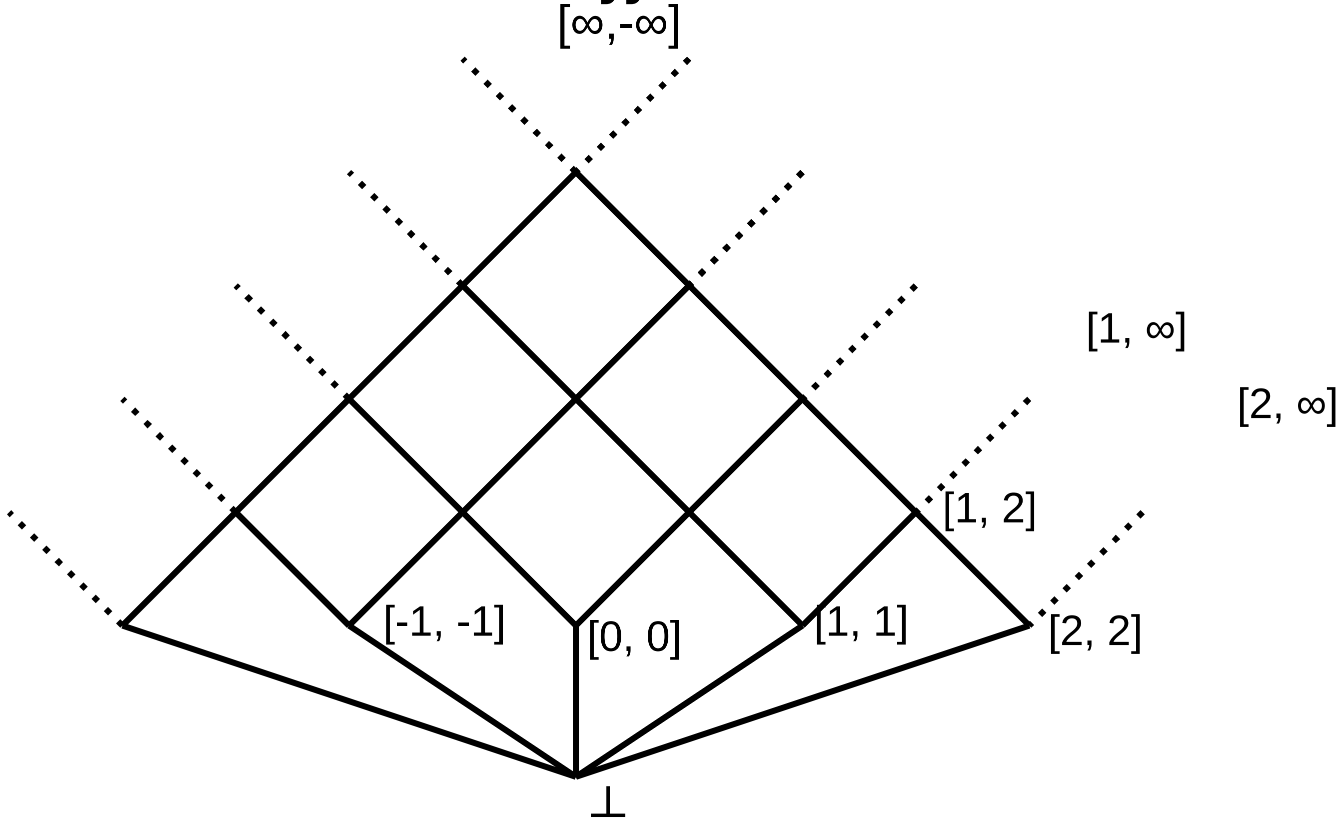
$\top$ = sup L (top)

$\bot$ = inf L (bottom)

Vollständiger Verband

Intervall = { ⊥ } oder {[z_1, z_2] | $z_1 \leq z_2$, z_1, $z_2 \in$ N,

N = {-∞, ..., -1, 0, 1, ..., ∞}}



Sei f : L → L eine Funktion auf einem vollständigen Verband L. Die Menge aller Fixpunkte:

$$\mathrm{Fix}(f) = \{ l \in L \mid f(l) = l \}$$

Die Menge aller Präfixpunkte :

$$Pre(f) = \{l \in L \mid f(l) \sqsubseteq l\}$$

Die Menge aller Postfixpunkte:

$$Post(f) = \{l \in L \mid f(l) \sqsupseteq l\}$$

Der kleinste Fixpunkt (least fixed point):

$$lfp(f) = \sqcap Fix(f)$$

Der größte Fixpunkt (greatest fixed point):

$$gfp(f) = \bigsqcup Fix(f)$$

Theorem von Knaster-Tarski

Sei (L, ⊑) ein vollständiger Verband und $f : L \to L$ eine monotone Funktion. Dann gilt:

$$lfp(f) = \sqcap Pre(f) \in Fix(f)$$

$$gfp(f) = \sqcup Post(f) \in Fix(f)$$

Folgerungen:

- jede monotone Funktion besitzt mind. einen Fixpunkt
- Die Menge aller Fixpunkte bildet einen vollständigen verband

Fixpunkt-Iteration

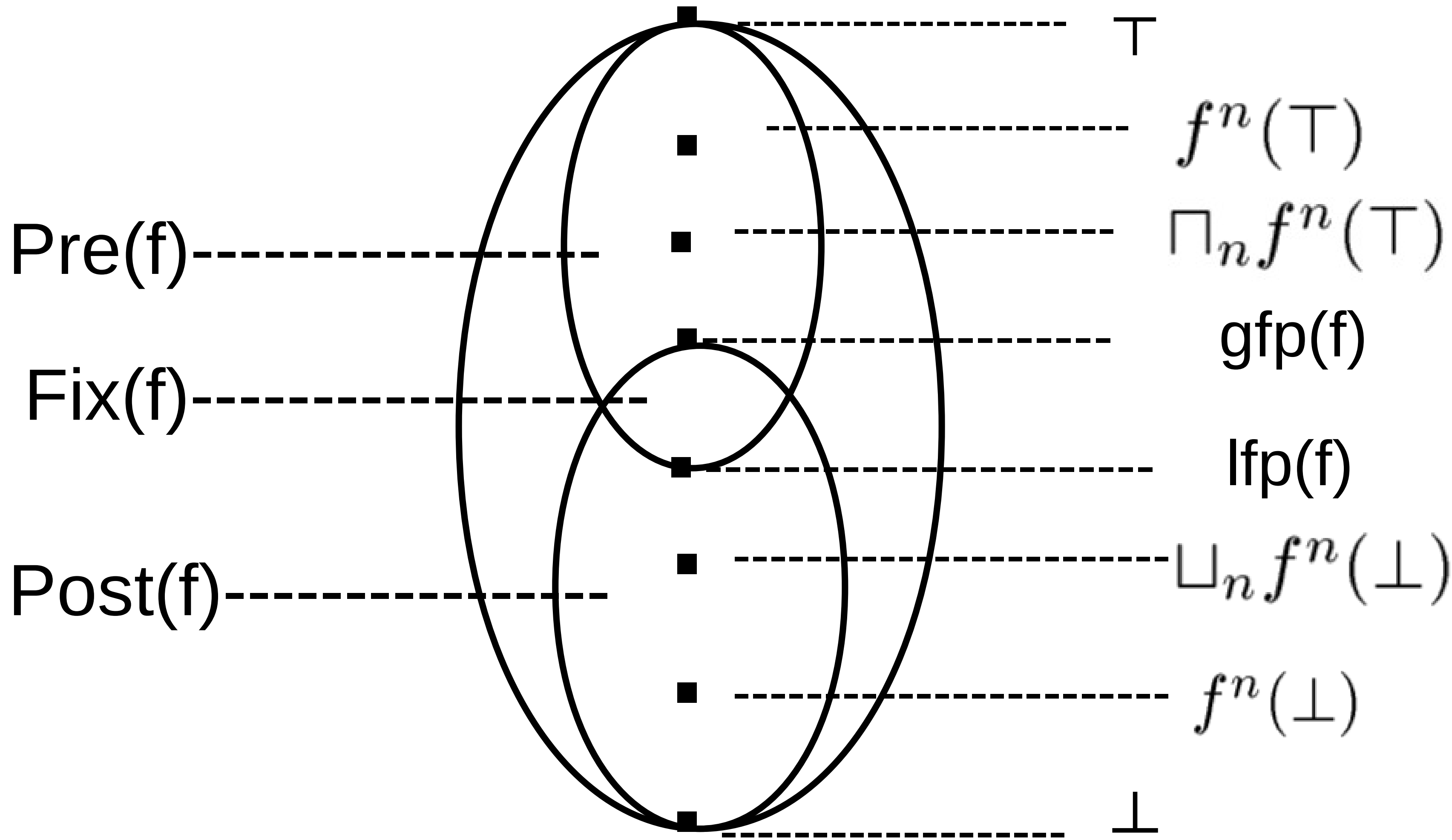
Sei f eine monotone Funktion auf L, nach dem Satz von Kleene gilt:

lfp(f) = $\bigsqcup_{n=0}^{\infty} f^n(\perp) = f^m(\perp)$

man berechnet $f(\perp) \bigsqcup f(f(\perp)) \bigsqcup ... \bigsqcup f'(\perp) \bigsqcup ...$

bis die Folge stationär ist und erhält dann den kleinsten Fixpunkt

Fixpunkte



$\top$
$f^n(\top)$
$\sqcap_n f^n(\top)$
gfp(f)
Pre(f)
Fix(f)
lfp(f)
$\sqcup_n f^n(\bot)$
Post(f)
$f^n(\bot)$
$\bot$

Upper Bound Operator

Operator $\uplus : L \times L \to L$ auf dem vollständigen Verband $L = (L, \sqsubseteq)$ heißt Upper Bound Operator, wenn:

$$l_1 \sqsubseteq (l_1 \uplus l_2) \sqsupseteq l_2$$

für alle $l1, l2 \in L$

Zurückgeliefert wird der Wert, der immer größer beider Argumente ist

Widening Operator

Operator $\nabla : \mathrm{L} \times \mathrm{L} \rightarrow \mathrm{L}$ auf dem vollständigen Verband ist ein Widening Operator nur dann, wenn:

- Das ist ein Upper Bound Operator, und
- die aufsteigende Kette $(l_n^{\nabla})_n$ wird letztendlich stabil für alle aufsteigenden Ketten $(l_n)_n$

Widening Operator auf einem Intervall:

$$[l_0, u_0] \nabla [l_1, u_1] = [\text{if } l_1 < l_0 \text{ then } -\infty \text{ else } l_0;\ \text{if } u_1 > u_0 \text{ then } +\infty \text{ else } u_0]$$

Widening Operator auf einem Intervall:

$[l_0, u_0] \nabla [l_1, u_1] = [\text{if } l_1 < l_0 \text{ then } -\infty \text{ else } l_0;$

$\text{if } u_1 > u_0 \text{ then } +\infty \text{ else } u_0]$

$[0,1] \nabla [0, 2] = [0, +\infty]$

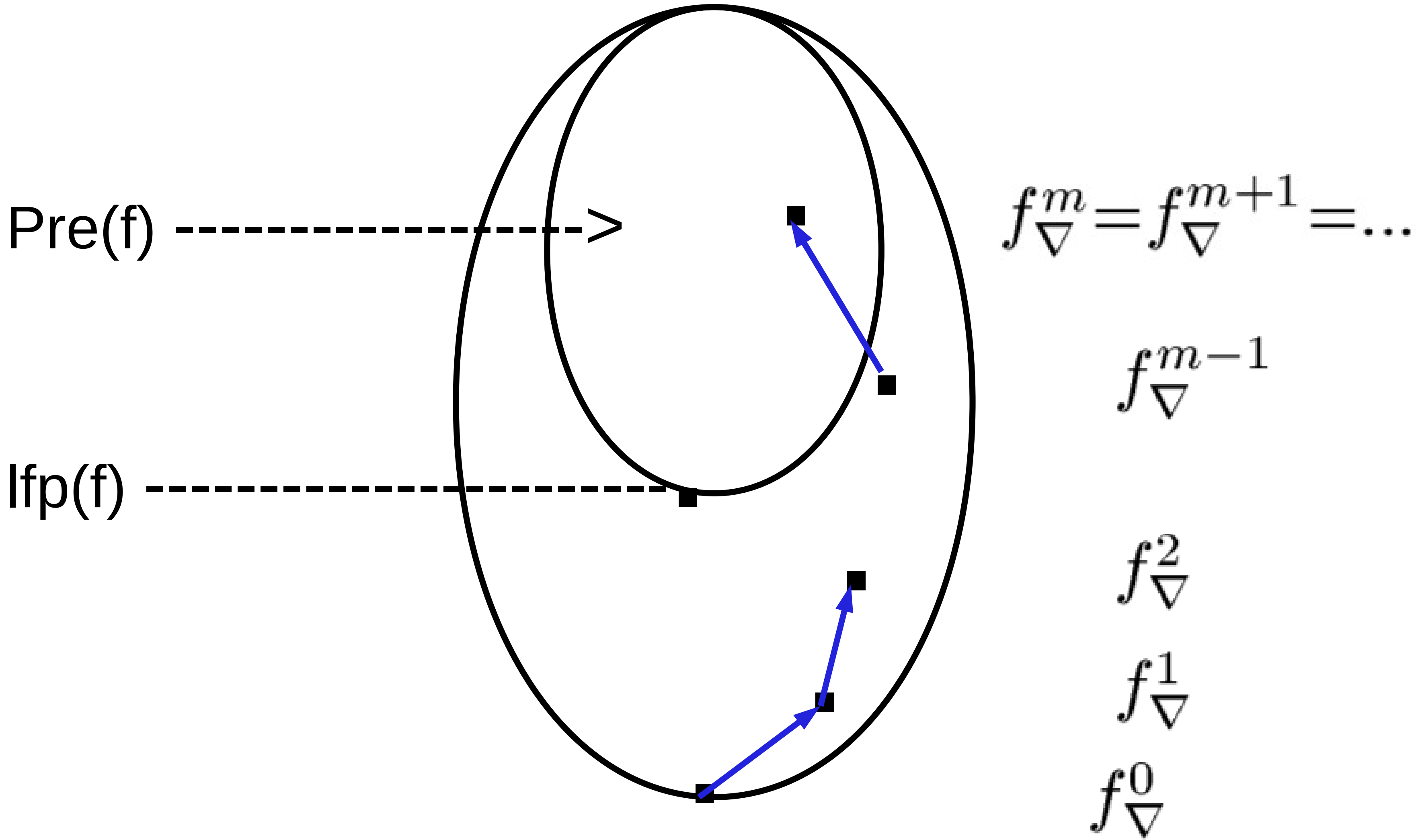
$[0,2] \nabla [0, 2] = [0, 2]$

Widening Operator

Mit dem Widening Operator und einer monotonen Funktion f : L x L → L kalkuliert man die Sequenz $(f_{\nabla}^{n})_n$

$$(f_{\nabla}^{n}) = \begin{cases} \perp & , falls\ n = 0 \\ (f_{\nabla}^{n} - 1) & , falls\ n\ \geq\ 0 \wedge f(f_{\nabla}^{n} - 1) \sqsubseteq (f_{\nabla}^{n} - 1) \\ (f_{\nabla}^{n} - 1) \nabla f(f_{\nabla}^{n} - 1) & , sonst \end{cases}$$

Widening Operator



Narrowing Operator

Operator Δ ist auf dem vollständigen Verband L ein Narrowing Operator nur dann, wenn

$$l_2 \sqsubseteq l_1 \rightarrow l_2 \sqsubseteq (l_1 \Delta l_2) \sqsupseteq l_1$$

für alle $\mathrm{l1}, \mathrm{l2} \in \mathrm{L}$ und

- für alle absteigenden Ketten $(l_n)_n$ wird die Sequenz $(l_n^{\nabla})_n$ letztendlich stabil

Narrowing Operator

- Nach Verwendung des Widening kann das Intervall sehr ungenau sein => Narrowing Operator versucht, die möglichen Werte wieder einzuschränken. Die Sequenz $(f_{\Delta}^{n})_n$ wird berechnet:

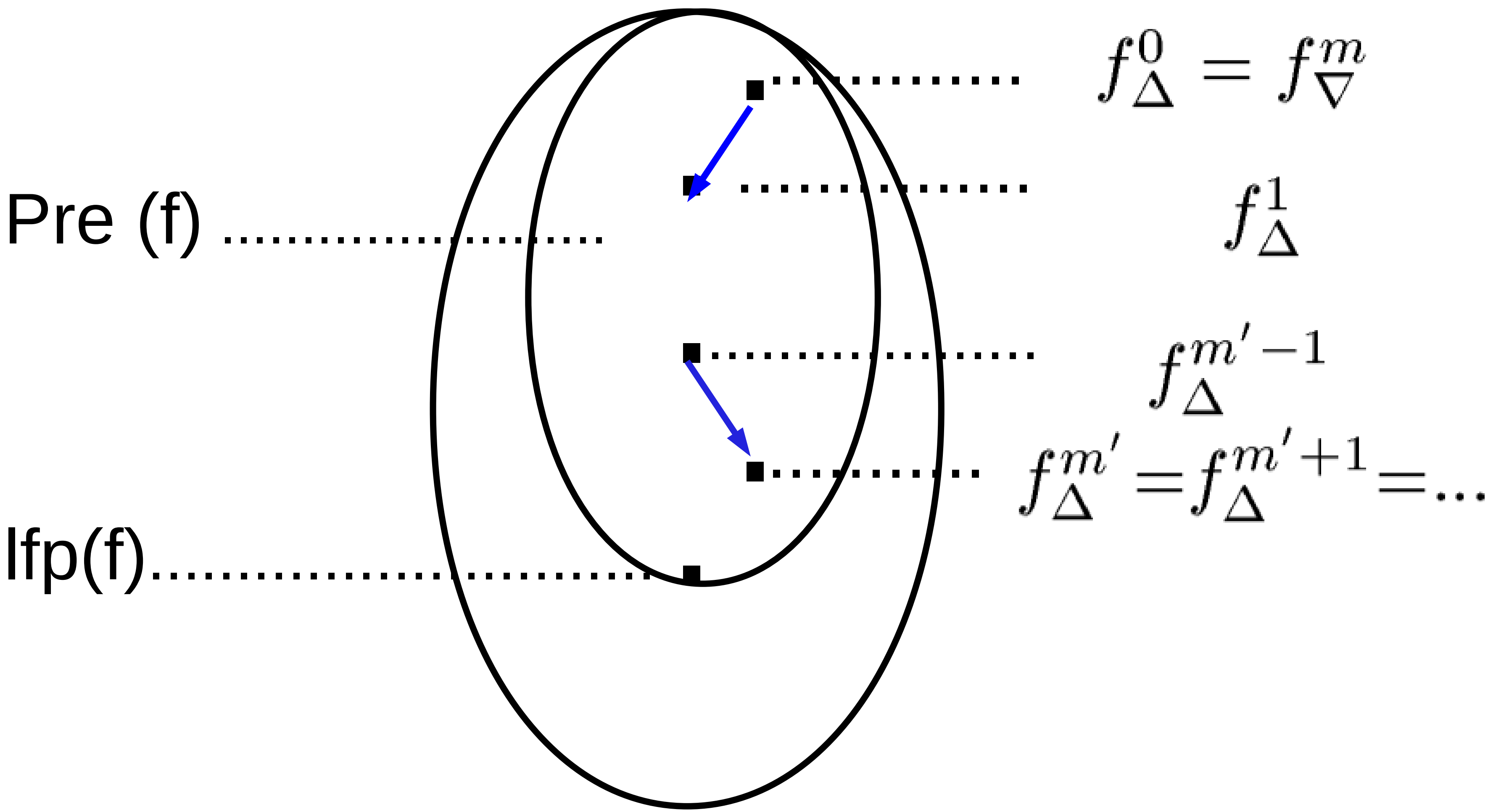
$$(f_{\Delta}^{n}) = \begin{cases} f_{\nabla}^{m} & , \; falls \; n = 0 \\ f_{\Delta}^{n-1} \; \Delta \; f(f_{\Delta}^{n-1}) & , \; falls \; n \; \geq \; 0 \end{cases}$$

Narrowing Operator

Folgerung: wenn Δ Narrowing Operator und $f(f_{\Delta}^{m}) \sqsubseteq f_{\Delta}^{m}$, dann

- $(f_{\Delta}^{n})_n$ eine absteigende Kette, und
- $f_{\Delta}^{n} \sqsupseteq f^{n}(f_{\Delta}^{m}) \sqsupseteq lfp(f)$

Narrowing Operator



Beispiel (1)


```
    x := 1;
1:
    while x < 10000 do
2:
    x := x+1
3:
    od;
4:
```

$$
\begin{cases}
X1 = [1, 1] \\
X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\
X3 = X2 \ominus [1, 1] \\
X4 = (X1 \sqcup X3) \cap [10000, \infty]
\end{cases}
$$

Beispiel (2)


```
    x := 1;
1:
    while x < 10000 do
2:
        x := x+1
3:
    od;
4:
```

$$\begin{cases} X1 = [1, 1] \\ X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\ X3 = X2 \ominus [1, 1] \\ X4 = (X1 \sqcup X3) \cap [10000, \infty] \end{cases}$$

$$\begin{cases} X1 = \emptyset \\ X2 = \emptyset \\ X3 = \emptyset \\ X4 = \emptyset \end{cases}$$

Beispiel (3)

```
    x := 1;
1:
    while x < 10000 do
2:
          x := x+1
3:
     od;
4:
```

$$
\begin{cases}
X1 = [1, 1] \\
X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\
X3 = X2 \ominus [1, 1] \\
X4 = (X1 \sqcup X3) \cap [10000, \infty]
\end{cases}
$$

$$
\begin{cases}
X1 = [1, 1] \\
X2 = \emptyset \\
X3 = \emptyset \\
X4 = \emptyset
\end{cases}
$$

Beispiel (4)


```
    x := 1;
1:
    while x < 10000 do
2:
          x := x+1
3:
    od;
4:
```

$$\begin{cases} X1 = [1, 1] \\ X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\ X3 = X2 \ominus [1, 1] \\ X4 = (X1 \sqcup X3) \cap [10000, \infty] \end{cases}$$

$$\begin{cases} X1 = [1, 1] \\ X2 = [1, 1] \\ X3 = \emptyset \\ X4 = \emptyset \end{cases}$$

Beispiel (5)


```
    x := 1;
1:
    while x < 10000 do
2:
          x := x+1
3:
    od;
4:
```

$$\begin{cases} X1 = [1, 1] \\ X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\ X3 = X2 \ominus [1, 1] \\ X4 = (X1 \sqcup X3) \cap [10000, \infty] \end{cases}$$

$$\begin{cases} X1 = [1, 1] \\ X2 = [1, 1] \\ X3 = [2, 2] \\ X4 = \emptyset \end{cases}$$

Beispiel (6)

```
    x := 1;
1:
    while x < 10000 do
2:
          x := x+1
3:
    od;
4:
```

$$\begin{cases} X1 = [1, 1] \\ X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\ X3 = X2 \ominus [1, 1] \\ X4 = (X1 \sqcup X3) \cap [10000, \infty] \end{cases}$$

$$\begin{cases} X1 = [1, 1] \\ X2 = [1, 2] \\ X3 = [2, 2] \\ X4 = \emptyset \end{cases}$$

Beispiel (7)

```
    x := 1;
1:
    while x < 10000 do
2:
          x := x+1
3:
    od;
4:
```

$$\begin{cases} X1 = [1, 1] \\ X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\ X3 = X2 \ominus [1, 1] \\ X4 = (X1 \sqcup X3) \cap [10000, \infty] \end{cases}$$

$$\begin{cases} X1 = [1, 1] \\ X2 = [1, 2] \\ X3 = [2, 3] \\ X4 = \emptyset \end{cases}$$

Beispiel (8)

```
    x := 1;
1:
    while x < 10000 do
2:
        x := x+1
3:
    od;
4:
```

$$\begin{cases} X1 = [1, 1] \\ X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\ X3 = X2 \ominus [1, 1] \\ X4 = (X1 \sqcup X3) \cap [10000, \infty] \end{cases}$$

$$\begin{cases} X1 = [1, 1] \\ X2 = [1, 3] \\ X3 = [2, 3] \\ X4 = \emptyset \end{cases}$$

Beispiel (9)

```
    x := 1;
1:
    while x < 10000 do
2:
          x := x+1
3:
    od;
4:
```

$$\begin{cases} X1 = [1, 1] \\ X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\ X3 = X2 \ominus [1, 1] \\ X4 = (X1 \sqcup X3) \cap [10000, \infty] \end{cases}$$

$$\begin{cases} X1 = [1, 1] \\ X2 = [1, 5] \\ X3 = [2, 6] \\ X4 = \emptyset \end{cases}$$

Beispiel (10)


```
    x := 1;
1:
    while x < 10000 do
2:
        x := x+1
3:
    od;
4:
```

$$\begin{cases} X1 = [1, 1] \\ X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\ X3 = X2 \ominus [1, 1] \\ X4 = (X1 \sqcup X3) \cap [10000, \infty] \end{cases}$$

$$\begin{cases} X1 = [1, 1] \\ X2 = [1, +\infty] \\ X3 = [2, 6] \\ X4 = \varnothing \end{cases}$$

Beispiel (11)


```
    x := 1;
1:
    while x < 10000 do
2:
          x := x+1
3:
    od;
4:
```

$$\begin{cases} X1 = [1, 1] \\ X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\ X3 = X2 \ominus [1, 1] \\ X4 = (X1 \sqcup X3) \cap [10000, \infty] \end{cases}$$

$$\begin{cases} X1 = [1, 1] \\ X2 = [1, +\infty] \\ X3 = [2, +\infty] \\ X4 = \emptyset \end{cases}$$

Beispiel (12)


```
    x := 1;
1:
    while x < 10000 do
2:
            x := x+1
3:
    od;
4:
```

$$\begin{cases} X1 = [1, 1] \\ X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\ X3 = X2 \ominus [1, 1] \\ X4 = (X1 \sqcup X3) \cap [10000, \infty] \end{cases}$$

$$\begin{cases} X1 = [1, 1] \\ X2 = [1, 9999] \\ X3 = [2, +\infty] \\ X4 = \emptyset \end{cases}$$

Beispiel (13)

```
    x := 1;
1:
    while x < 10000 do
2:
        x := x+1
3:
    od;
4:
```

$$\begin{cases} X1 = [1, 1] \\ X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\ X3 = X2 \ominus [1, 1] \\ X4 = (X1 \sqcup X3) \cap [10000, \infty] \end{cases}$$

$$\begin{cases} X1 = [1, 1] \\ X2 = [1, 9999] \\ X3 = [2, +10000] \\ X4 = \emptyset \end{cases}$$

Beispiel (14)

```
    x := 1;
1:
    while x < 10000 do
2:
            x := x+1
3:
    od;
4:
```

$$\begin{cases} X1 = [1, 1] \\ X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\ X3 = X2 \ominus [1, 1] \\ X4 = (X1 \sqcup X3) \cap [10000, \infty] \end{cases}$$

$$\begin{cases} X1 = [1, 1] \\ X2 = [1, 9999] \\ X3 = [2, +10000] \\ X4 = [+10000, +10000] \end{cases}$$

Beispiel (15)

x := 1;

1: {x = 1}

while x < 10000 do

2: {x ∈ [1, 9999]}

x := x+1

3: {x ∈ [2, 10000]}

od;

4: {x = 10000}

$$\begin{cases} X1 = [1, 1] \\ X2 = (X1 \sqcup X3) \cap [-\infty, 9999] \\ X3 = X2 \ominus [1, 1] \\ X4 = (X1 \sqcup X3) \cap [10000, \infty] \end{cases}$$

$$\begin{cases} X1 = [1, 1] \\ X2 = [1, 9999] \\ X3 = [2, +10000] \\ X4 = [10000, 10000] \end{cases}$$

III.Zusammenfassung