

LMSE

Logische Methoden des Software
Engineerings

Prof. Dr. Jakob Rehof

Lehrstuhl XIV, Software Engineering

Fundamentale Modelle

1930'er

Rekursive Funktionen (Gödel)	Turing Maschine (Turing)	Lambda Kalkül (Church)
------------------------------------	--------------------------------	------------------------------

Fundamentale Modelle

1936-37

Äquivalenz der Modelle

- Kleene-Rosser (1936): alle rekursive Funktionen sind im Lambda-Kalkül repräsentierbar
- Turing (1937): Genau die Funktionen, die mit einer Turing-Maschine berechenbar sind, sind im Lambda-Kalkül repräsentierbar

1930'er

Rekursive Funktionen (Gödel)	Turing- Maschine (Turing)	Lambda- Kalkül (Church)
------------------------------------	---------------------------------	-------------------------------

Fundamentale Modelle des Rechnens

1936-37 Äquivalenz der Modelle

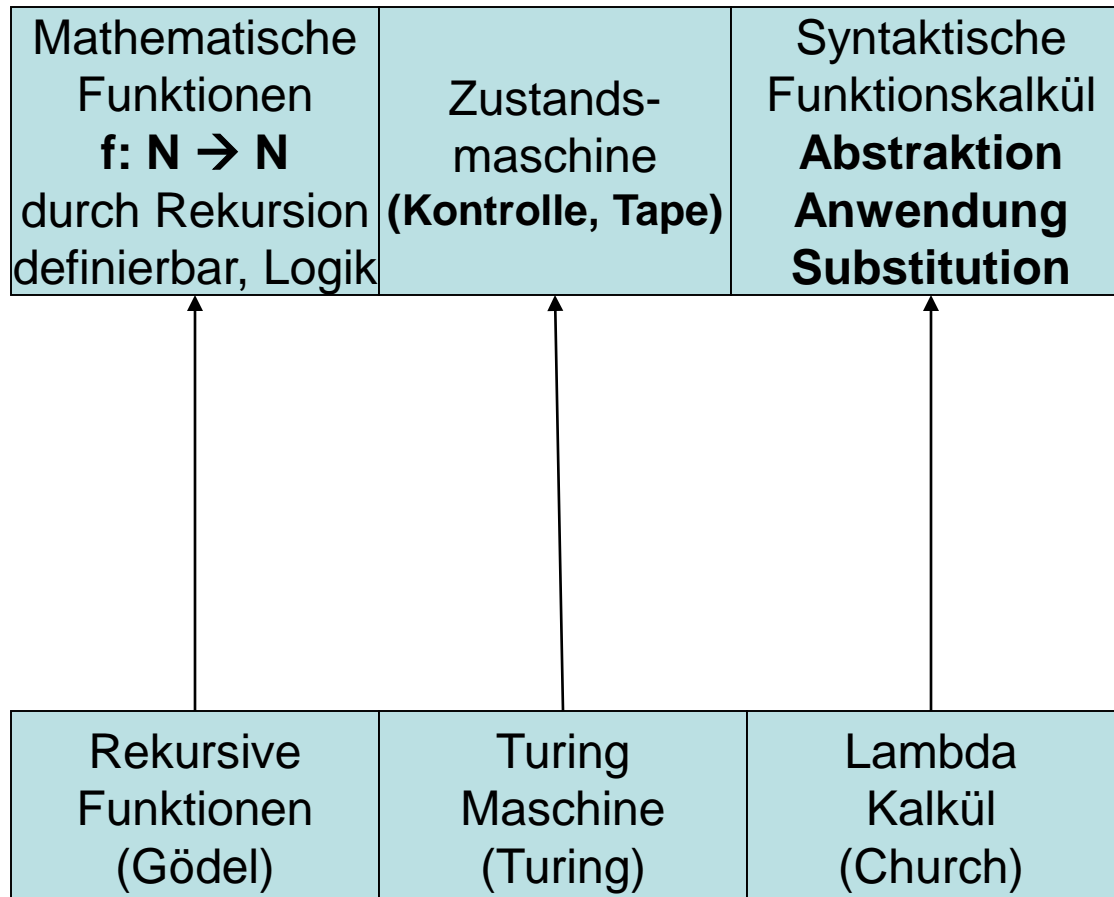
- Kleene-Rosser (1936): alle rekursive Funktionen sind im Lambda-Kalkül repräsentierbar
- Turing (1937): Genau die Funktionen, die mit einer Turing-Maschine berechenbar sind, sind im Lambda-Kalkül repräsentierbar

Turing-Vollständigkeit Church-Turing Thesis

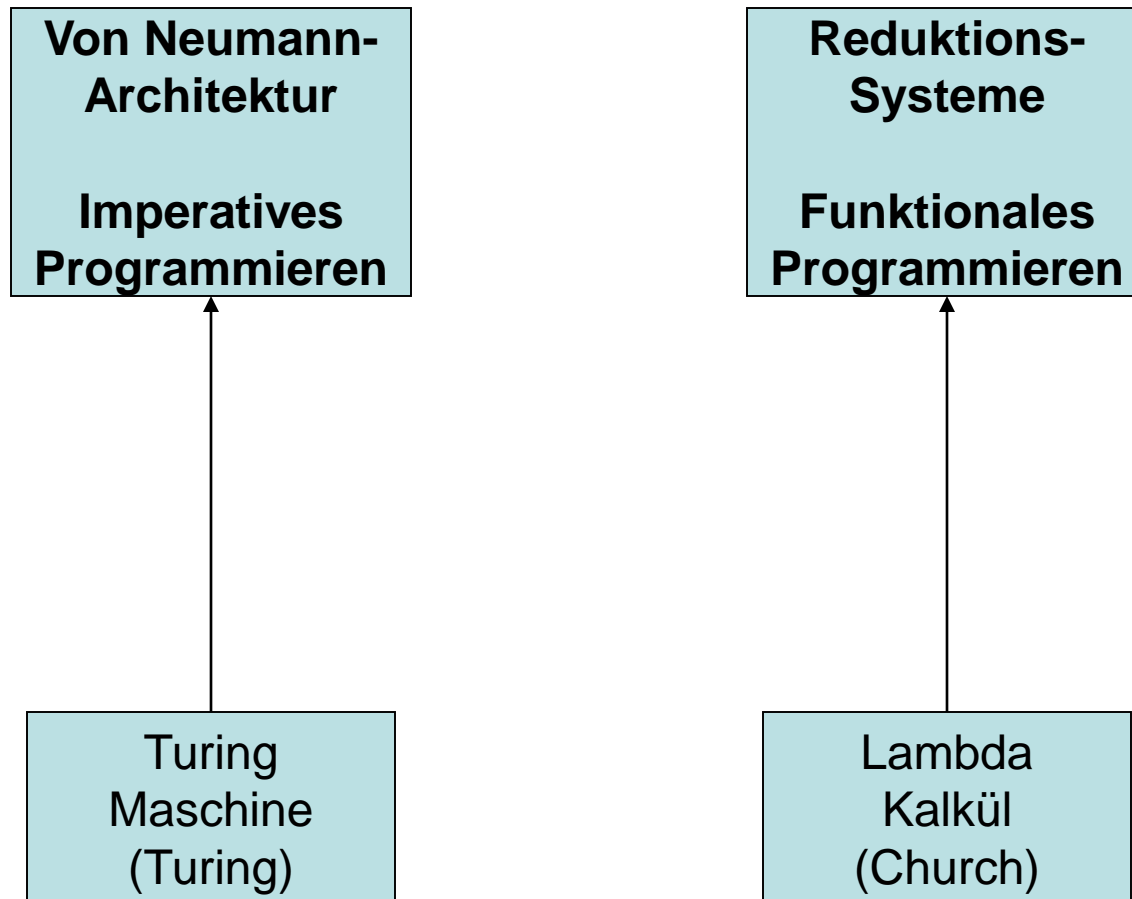
1930'er

Rekursive Funktionen (Gödel)	Turing-Maschine (Turing)	Lambda-Kalkül (Church)
---------------------------------	-----------------------------	---------------------------

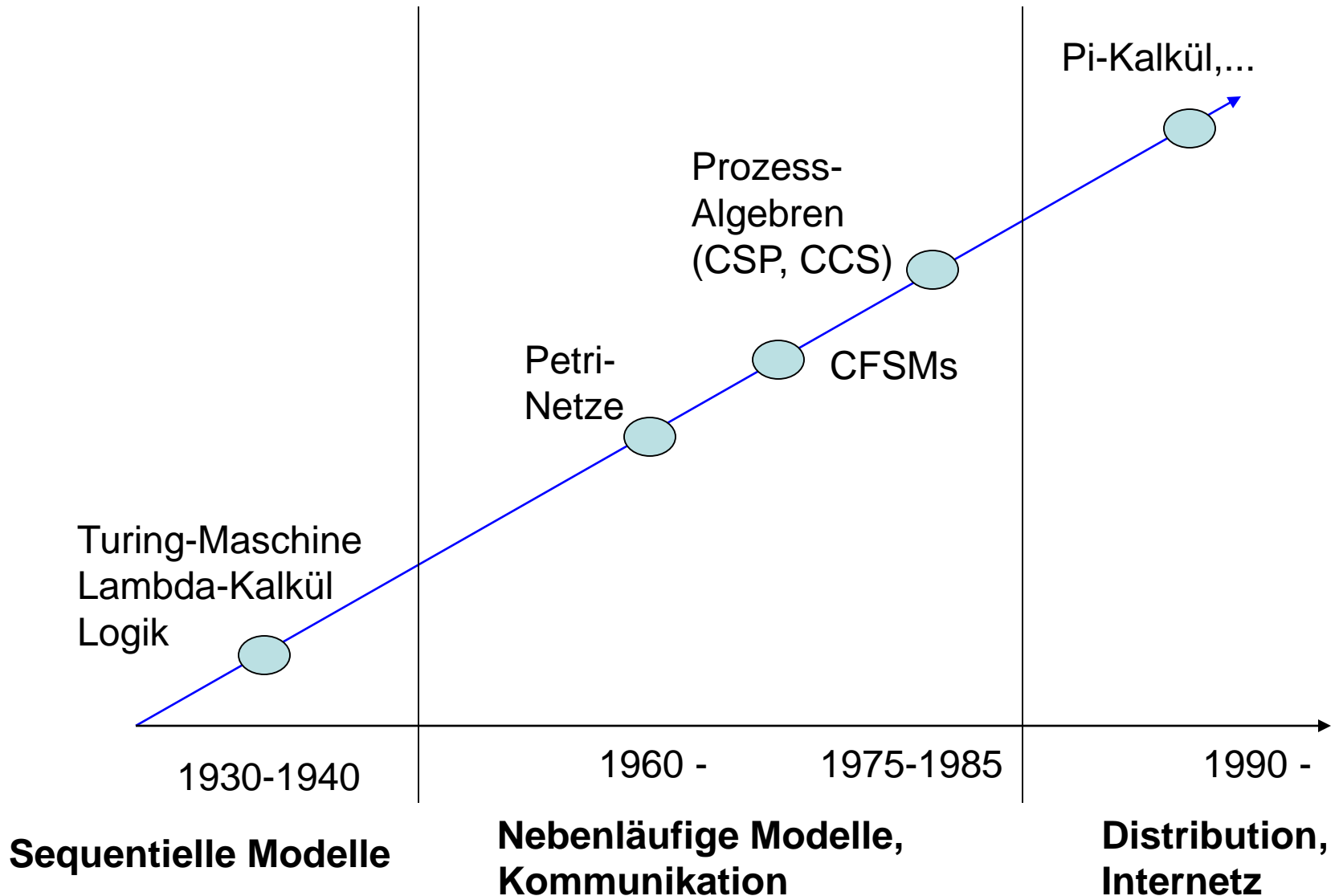
Fundamentale Begriffe (Reduktionen)



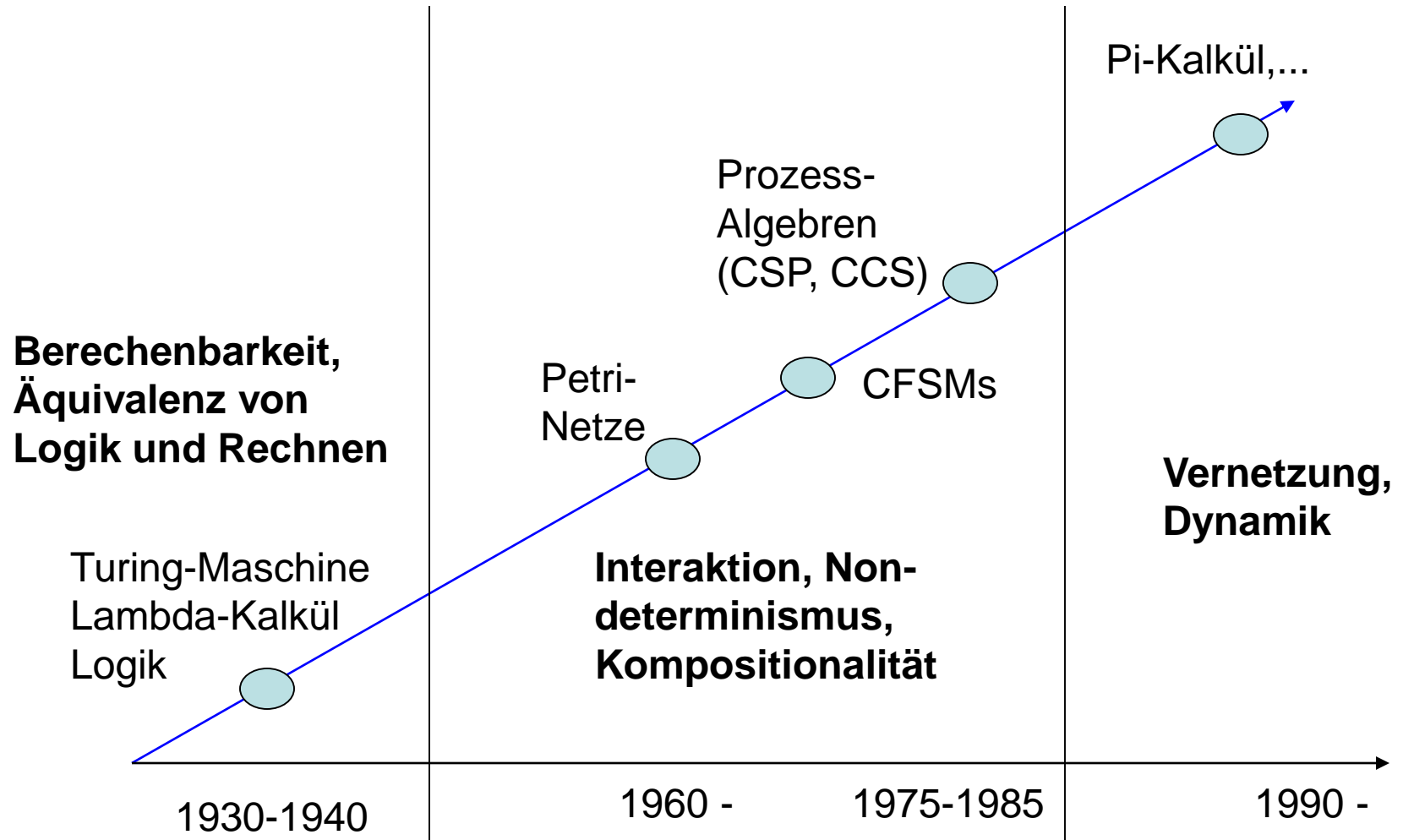
Fundamentale Architekturen



Fundamentale Modelle



Fundamentale Erkenntnisse



Lambda-Kalkül

- Definition von Funktionsausdrücken
(Abstraktion)

$$\lambda x. M$$

- **Anwendung** von Funktion auf Argument

$$(F A)$$

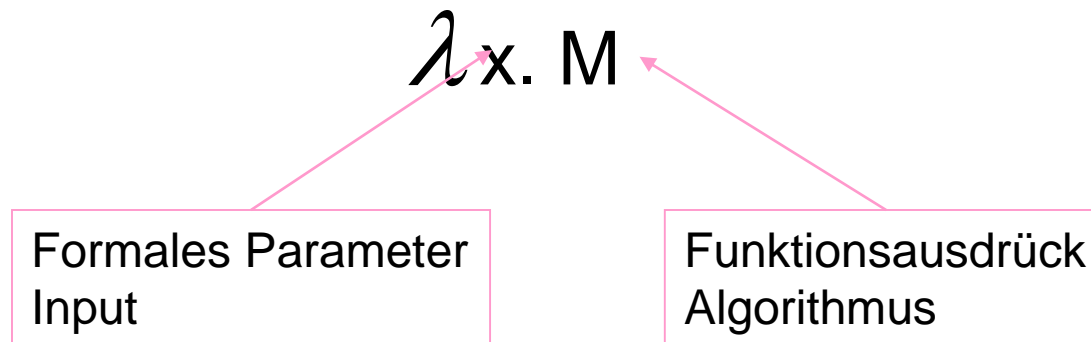
- **Rechenvorschrift** für Anwendungen

Notationssystem

$\lambda x. M$: Notationssystem für
anonyme Funktionsdefinitionen, sowie
„ $x \mapsto M(x)$ “

Notationssystem

$\lambda x. M$: Notationssystem für
anonyme Funktionsdefinitionen, sowie
„ $x \mapsto M(x)$ “



Lambda Kalkül

- Syntaktische Funktionen
- Reiner vs Typorientierter Lambda-Kalkül
- Typentheorie
- Konstruktive Logik
- Funktionales Programmieren
- CPS-Transformation
- ...
- LSXIV: Kompositionssynthese

Lambda-Kalkül (rein, ohne Typen)

- Rein (pure): keine Funktionsprimitiven
 - d.h., alles muss definiert/kodiert werden
- Ohne Typen (type-free): alle Kombinationen von Funktionen und Argumenten sind erlaubt
 - unbegrenzte Selbstanwendungen (F F)

Getyptes Lambda-Kalkül (Simple typed lambda calculus)

$\lambda \rightarrow$

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \rightarrow \tau}$$

$$\frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash M N : \tau}$$

Konstruktive (intuitionistische) Aussagenlogik

$\Gamma, \varphi \vdash \varphi$ (Ax)

$$\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi} (\wedge I)$$

$$\frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi} (\wedge E) \quad \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \psi}$$

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi} (\vee I) \quad \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi}$$

$$\frac{\Gamma, \varphi \vdash \rho \quad \Gamma, \psi \vdash \rho}{\Gamma \vdash \rho} \quad \frac{\Gamma \vdash \varphi \vee \psi}{\Gamma \vdash \rho} (\vee E)$$

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \rightarrow \psi} (\rightarrow I)$$

$$\frac{\Gamma \vdash \varphi \rightarrow \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi} (\rightarrow E)$$

$$\frac{\Gamma \vdash \perp}{\Gamma \vdash \varphi} (\perp E)$$

Curry-Howard Isomorphism

$\lambda \rightarrow$

IPC(\rightarrow)

term variable

assumption

term

construction (proof)

type variable

propositional variable

type

formula

type constructor

connective

inhabitation

provability

typable term

construction for a proposition

redex

construction representing proof tree with redundancy

reduction

normalization

value

normal construction