

# Aby Spec von SW & HW II

3.1.  $\Delta(\text{RAT}) = \mathbb{Q}$

$$\Delta(0) = 0$$

$$\Delta(1) = 1$$

$$\Delta(\_ \cdot \_)(x, y) = x \cdot y \quad \text{für alle } x, y \in \Delta(\text{RAT})$$

$$\Delta(\_ / \_)(x, y) = \begin{cases} x/y & y \neq 0 \\ \perp & \text{sonst} \end{cases} \quad \text{für alle } x, y \in \Delta(\text{RAT})$$

$$\Delta(\text{isNat}) = \mathbb{N}$$

3.2.  $\Delta(\text{RAT}) = \mathbb{Z}$

$$\Delta(0) = 0$$

$$\Delta(1) = 1$$

$$\Delta(\_ \cdot \_)(x, y) = x \cdot y \quad \text{für alle } x, y \in \Delta(\text{RAT})$$

$$\Delta(\_ / \_)(x, y) = \begin{cases} x/y = z & \text{für alle } x, y, z \in \Delta(\text{RAT}), y \neq 0 \\ \perp & \text{sonst} \end{cases}$$

$$\Delta(\text{isNat}) = \mathbb{N}$$

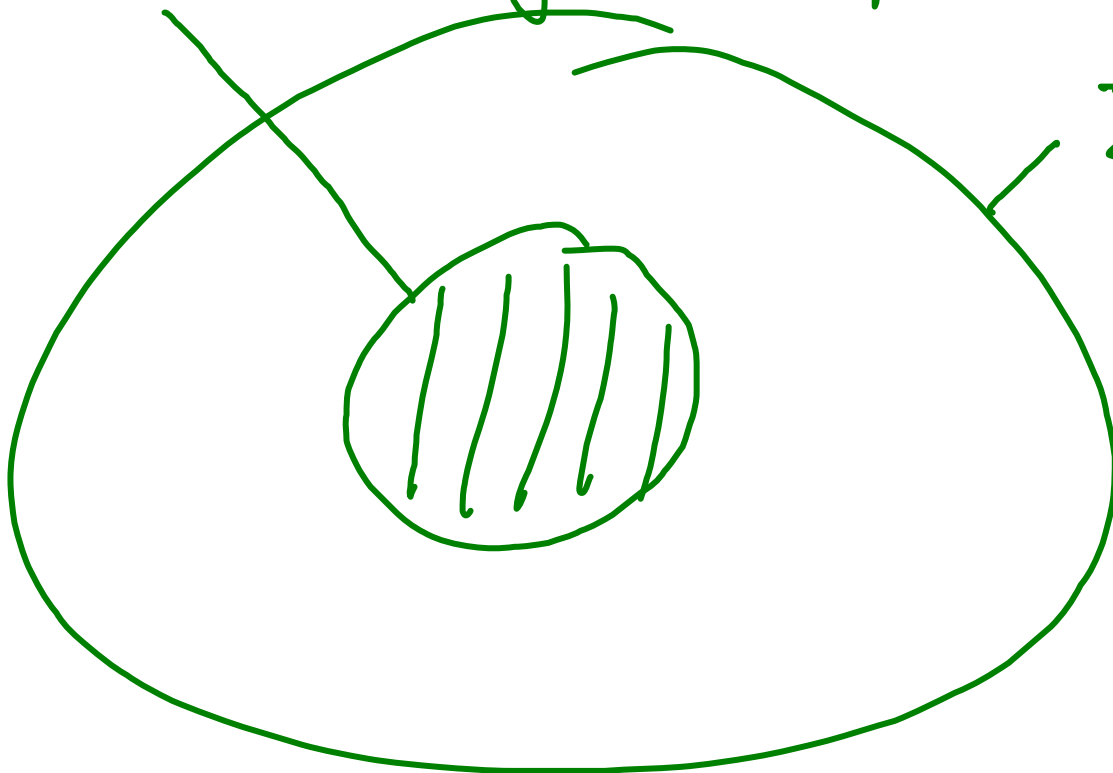


Menge von  
Eigenschaften

$\Sigma$

Symbole

$\Phi$ : Formeln in Logik 1. Stufe



$\Sigma$ -Algebren

"geben  
Interpretation  
der  
Symbole"

gegebenes Programm (Java)

$\approx$

Algebra

Wie kann ich das in CAS  
codieren?

gegeben in  $E$  Eigenschaft:

"kommutativ"

gilt diese Eigenschaft für  
mein Programm?

→ view

Hats kann es hoffentlich beantwortet

set  $\rho, \pi$

$$X = (X_\rho, X_\pi)$$

$$X_\rho \cap X_\pi = \emptyset$$

$$X_\rho = \{a, b\} \quad X_\pi = \{c\}$$

$$X = (X_{Nat}, X_{List})$$

$$X_{Nat} = \{n, m\}$$

$$X_{List} = \{l\}$$

-  $\text{succ}(0) ::= \text{nil}$

-  $\text{take}(0, \text{nil}) ++ \text{take}(\text{succ}(0), \text{nil})$

- 0



$X = (X_{\text{List}}, X_{\text{Nat}})$

$X_{\text{Nat}} \ni n$

$\text{succ}(n)$

detern. autom.  $A$

$$A = (Q, i, \Sigma, F, \delta)$$

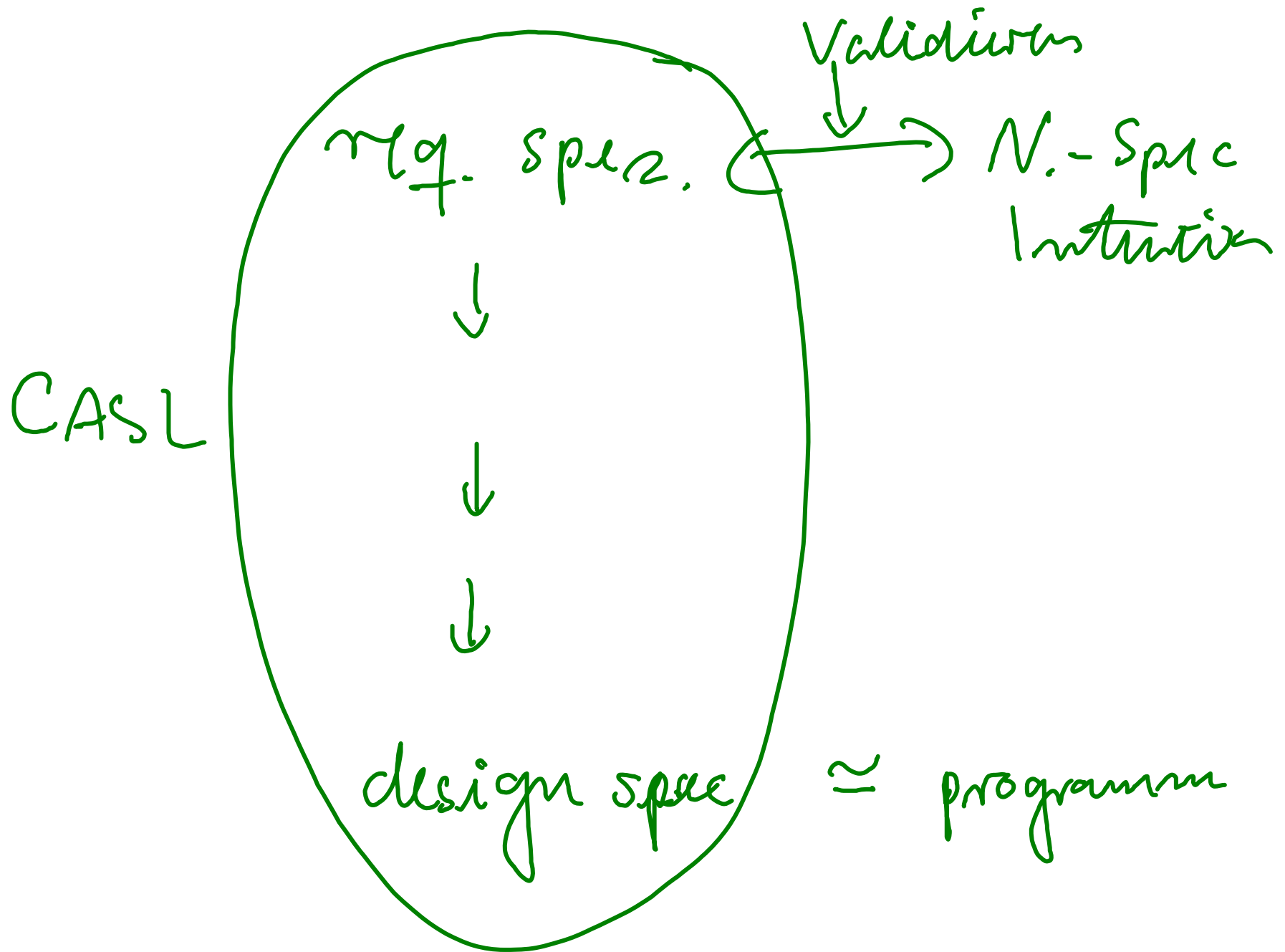
$Q$ : eine Menge von Zuständen

$i \in Q$ : start-Zustand

$\Sigma$ : Eingabealphabet

$F \subseteq Q$ : akzeptierenden Zustände

$\delta: Q \times \Sigma \rightarrow Q$



1.  $\forall m: \text{Move}, c: \text{Configuration}$ .

$\text{not movesNext}(\text{nextConfiguration}(m, c)) = \text{movesNext}(c)$

2.  $\forall m: \text{Move}, c: \text{Configuration}, p: \text{Player}$ .

$m \text{ is legal Move in } c \text{ for } p \Rightarrow \text{movesNext}(c) = p$

$\text{not movesNext}(c) = p \Rightarrow \text{not}_m \text{ is legal Move in } c \text{ for } p$

3.  $\forall c: \text{Configuration}, p_1: \text{Player}, p_2: \text{Player}$

$c \text{ is winning situation for } p_1 \wedge c \text{ is winning } \dots p_2$

$\Rightarrow p_1 = p_2$

4.  $\forall c: \text{Configuration}, p: \text{Player}, m: \text{Move}$

C is Winning Situation For A

$\Rightarrow \forall p: \text{Player } p \text{ is not a legal move in C for A}$

$$A(\text{state}) = \{A(s_0), A(s_1), A(s_2)\}$$

$$A(s_0) \in A(\text{is Final State})$$

$$\begin{aligned} \{A(s_1)\} &= A_1 \\ \{A(s_1), A(s_2)\} &= A_2(\text{is Final State}) \end{aligned}$$

$$\begin{aligned} \{A(s_1), \\ A(s_2), \\ A(s_3)\} &= A_3( \quad ) \end{aligned}$$