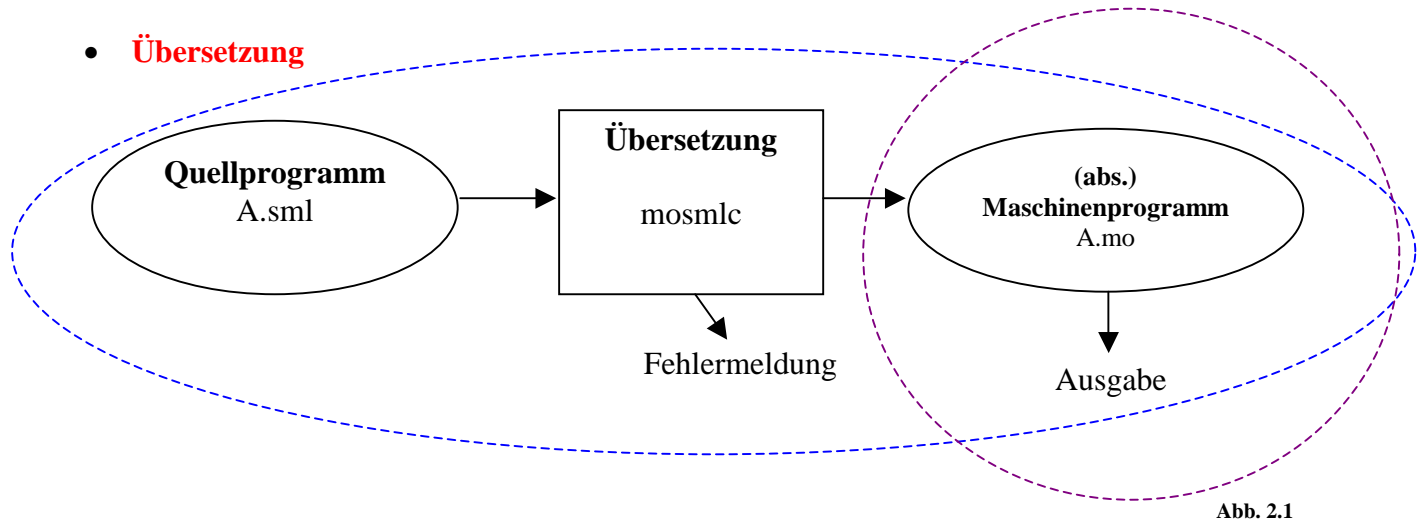


2. Implementierung von Programmiersprachen

2.1 Übersetzung und Implementierung

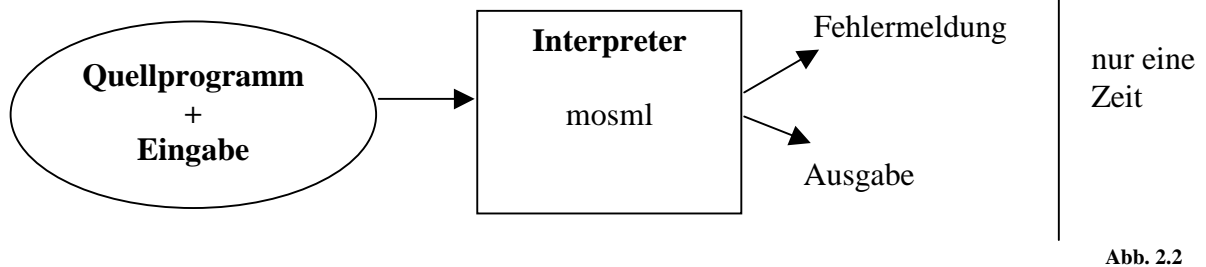
- **Übersetzung**



2 Zeiten:

1. Übersetzungszeit
2. Laufzeit

- **Interpretation**



- **Interpretation in einem interaktiven Modus**

Programm wird stückweise eingegeben

“Stück“ ist eine Folge von Deklarationen oder ein Ausdruck

Stück wird durch “; ↵” abgeschlossen

Interpreter liest Stück aus, wertet es in der aktuellen Arbeitsumgebung aus und verändert eventuell die Umgebung.

2.2 Struktur von Übersetzern / Interpretern

Übersetzer:

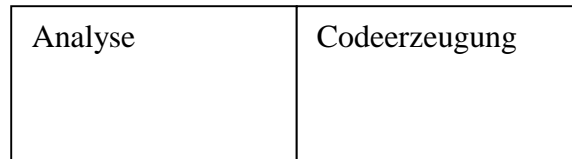


Abb 2.3

Interpreter:

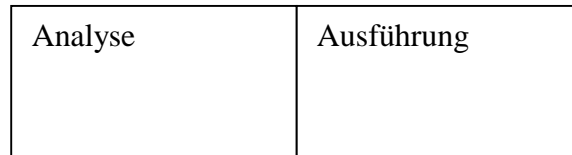


Abb 2.4

2.2.1 Aufgaben und Struktur der Analysephase

Die Analyse stellt fest,

- ob das eingegeben Programm(stück) korrekt aufgebaut ist (lexikalische, syntaktische Analyse)
- ob es typkorrekt ist (Typüberprüfung)
- lexikalische Analyse
 - Eingabe: Programmstück als Folge von Zeichen (ASCII)
 - Fasst Teilfolgen zu Symbolen (token) zusammen und stellt fest, zu welcher Symbolklasse die Teilfolge gehört
 - ❖ Name (identifizier)
 - ❖ Konstante
 - ❖ reservierte Worte (keywords)

Anmerkungen zur Moscow ML Language Overview

1. lexikalische Analyse

Programm(stück) als Folge von Zeichen

→ Folge von Symbolen

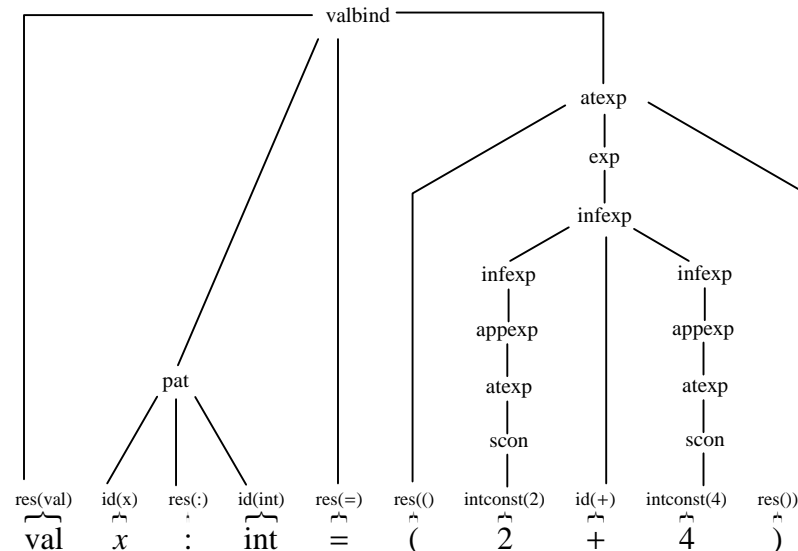
Symbole aus Symbolklassen

- Namen (identifier)
 - symbolische
 - alphanumerische

Folge von Buchstaben, Ziffern, einfachen Apostrophen und Unterstriche, beginnend mit einem Buchstaben oder einem Apostroph
- Spezialkonstanten
- reservierte Wörter: andalso, else, end, if, val

2. syntaktische Analyse

Zerlegt Symbolfolgen in ihre syntaktische Struktur (gemäß Grammatik). Stellt Struktur geeignet dar, z.B. durch einen Baum.



3. Typüberprüfung

Prüft,

- ob Operatoren zu den Typen der Operanden passen
- bei if e_1 then e_2 else e_3
ob e_1 den Typ `bool` hat und ob Typ e_2 gleich Typ e_3 ist
- Bei Funktionsanwendung $e_1 e_2$ ob e_1 Funktionstyp $t_1 \rightarrow t_2$ hat und ob e_2 den Typ t_1 hat