

# Compilerbau I

## Teil 9

**Definition:** Eine c.f. Grammatik  $G = (N, T, P, \sigma)$  heißt links-rekursiv  $\Leftrightarrow$

$$\exists \alpha \in N, u \in V^*: \alpha \rightarrow^+ \alpha u$$

Beispiel:

$$E \rightarrow E + T \mid T;$$

$$T \rightarrow T * F \mid F;$$

$$F \rightarrow ( E ) \mid \mathbf{ident}$$

$$\text{First}(T) \subseteq \text{First}(E) \subseteq \text{First}(E+T)$$

$$\Rightarrow \text{First}(T) \cap \text{First}(E+T) \neq \emptyset$$

$$\Rightarrow G \text{ ist nicht LL}(1)$$

**Definition:** Eine c.f. Grammatik  $G = (N, T, P, \sigma)$  heißt links-rekursiv  $\Leftrightarrow$

$$\exists \alpha \in N, u \in V^*: \alpha \rightarrow^+ \alpha u$$

**Transformation** zur Eliminierung (direkter) Links-Rekursion:

Sei  $G = (N, T, P, \sigma)$  eine c.f. Grammatik und  $\alpha \in N$  mit links-rekursiven Regeln:

$$\alpha \rightarrow \alpha u_1 \mid \alpha u_2 \mid \dots \mid \alpha u_m \mid v_1 \mid \dots \mid v_k \quad \text{mit: } u_i, v_j \in V^*, k > 0$$

kein  $v_j$  beginnt mit  $\alpha$

Wähle **neues Nichtterminal**  $\alpha'$  und **neue Produktionen:**

$$\alpha \rightarrow v_1 \alpha' \mid \dots \mid v_k \alpha' \quad \text{und} \quad \alpha' \rightarrow u_1 \alpha' \mid \dots \mid u_m \alpha' \mid \varepsilon$$

Ableitung in  $G$ :  $\alpha \rightarrow \alpha u_{i_1} \rightarrow \alpha u_{i_2} u_{i_1} \dots \rightarrow \alpha u_{i_n} \dots u_{i_1} \rightarrow v_j u_{i_n} \dots u_{i_1}$

Ableitung in  $G'$ :  $\alpha \rightarrow v_j \alpha' \rightarrow v_j u_{i_n} \alpha' \dots \rightarrow v_j u_{i_n} \dots u_{i_1} \alpha' \rightarrow v_j u_{i_n} \dots u_{i_1}$

## Beispiel für Elimination von Links-Rekursion

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow ( E ) \mid \mathbf{ident}$$
$$E \rightarrow TE' ; \quad E' \rightarrow + TE' \mid \varepsilon$$
$$T \rightarrow FT' ; \quad T' \rightarrow * FT' \mid \varepsilon ;$$
$$F \rightarrow ( E ) \mid \mathbf{ident}$$

**Transformation** zur Eliminierung (direkter) Links-Rekursion:

Sei  $G = (N, T, P, \sigma)$  eine c.f. Grammatik und  $\alpha \in N$  mit links-rekursiven Regeln:

$$\alpha \rightarrow \alpha u_1 \mid \alpha u_2 \mid \dots \mid \alpha u_m \mid v_1 \mid \dots \mid v_k \quad \text{mit: } u_i, v_j \in V^*$$

kein  $v_j$  beginnt mit  $\alpha$

Wähle **neues Nichtterminal**  $\alpha'$  und **neue Produktionen**:

$$\alpha \rightarrow v_1 \alpha' \mid \dots \mid v_k \alpha' \quad \text{und}$$
$$\alpha' \rightarrow u_1 \alpha' \mid \dots \mid u_m \alpha' \mid \varepsilon$$

## Beispiel für Elimination von Links-Rekursion

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow ( E ) \mid \mathbf{ident}$$
$$E \rightarrow TE' ; \quad E' \rightarrow + TE' \mid \varepsilon$$
$$T \rightarrow FT' ; \quad T' \rightarrow * FT' \mid \varepsilon ;$$
$$F \rightarrow ( E ) \mid \mathbf{ident}$$
$$\text{First}(E) = \text{First}(T) = \text{First}(F) = \{ (, \mathbf{ident} \}$$
$$\text{First}(E') = \{ +, \varepsilon \}; \text{First}(T') = \{ *, \varepsilon \}$$
$$\text{Follow}(E) = \text{Follow}(E') = \{ ), \# \}$$
$$\text{Follow}(T) = \text{Follow}(T') = \{ +, ), \# \}$$
$$\text{Follow}(F) = \{ +, *, ), \# \}$$

Kriterien für LL(1):

$$\text{First}(+TE') = \{ + \}; \text{First}(\varepsilon) = \{ \varepsilon \} \Rightarrow \text{First}(+TE') \cap \text{First}(\varepsilon) = \emptyset$$

ebenso: für die rechten Seiten von T' und F

$$\varepsilon \in \text{First}(E'): \quad \text{First}(E') \cap \text{Follow}(E') = \emptyset$$
$$\varepsilon \in \text{First}(T'): \quad \text{First}(T') \cap \text{Follow}(T') = \emptyset$$

$\Rightarrow G'$  ist LL(1)

stmnt  $\rightarrow$  **if** exp **then** stmnt

| **if** exp **then** stmnt **else** stmnt

stmnt  $\rightarrow$  **if** exp **then** stmnt if\_rest

if\_rest  $\rightarrow$  **else** stmnt |  $\epsilon$

Gemeinsamer Links-Faktor u:

$\alpha \rightarrow uu_1 \mid uu_2$  und  $\text{First}(u) \neq \emptyset$

**Transformation:**

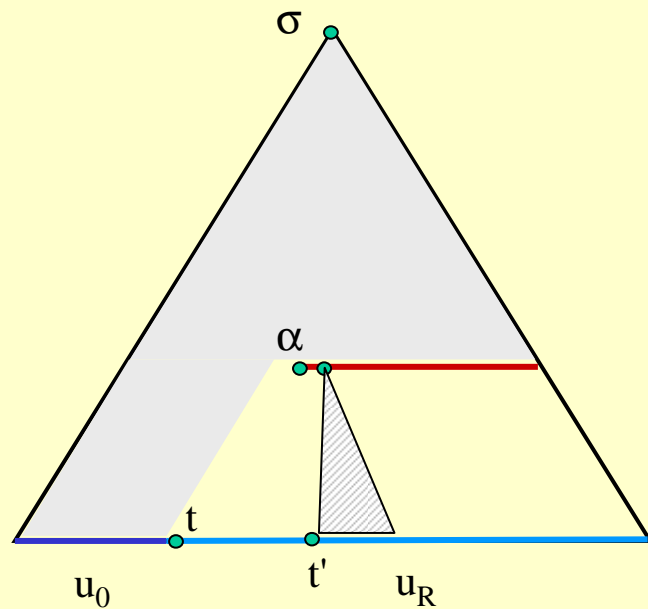
$\alpha \rightarrow u \alpha'$  und  $\alpha' \rightarrow u_1 \mid u_2$

## Synchronisierende Token

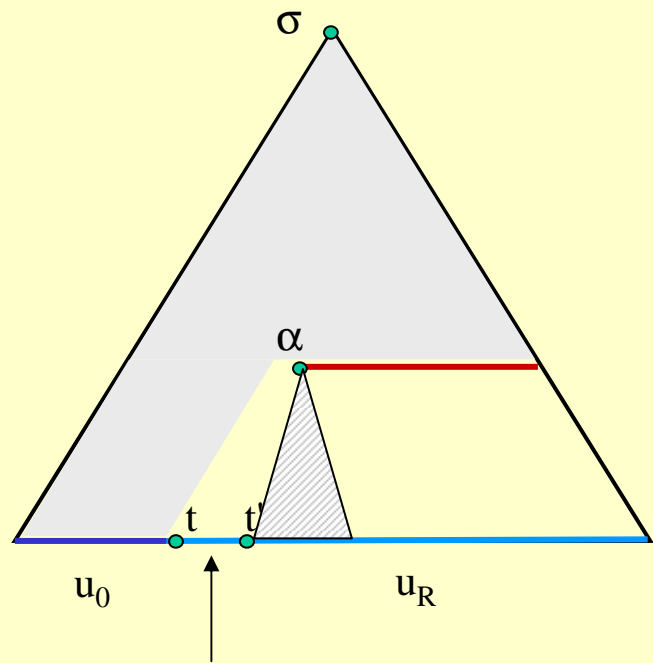
**Bei Fehler:** Eingabe bis zum nächsten synchronisierenden Token fortschreiten

### Panikreaktion:

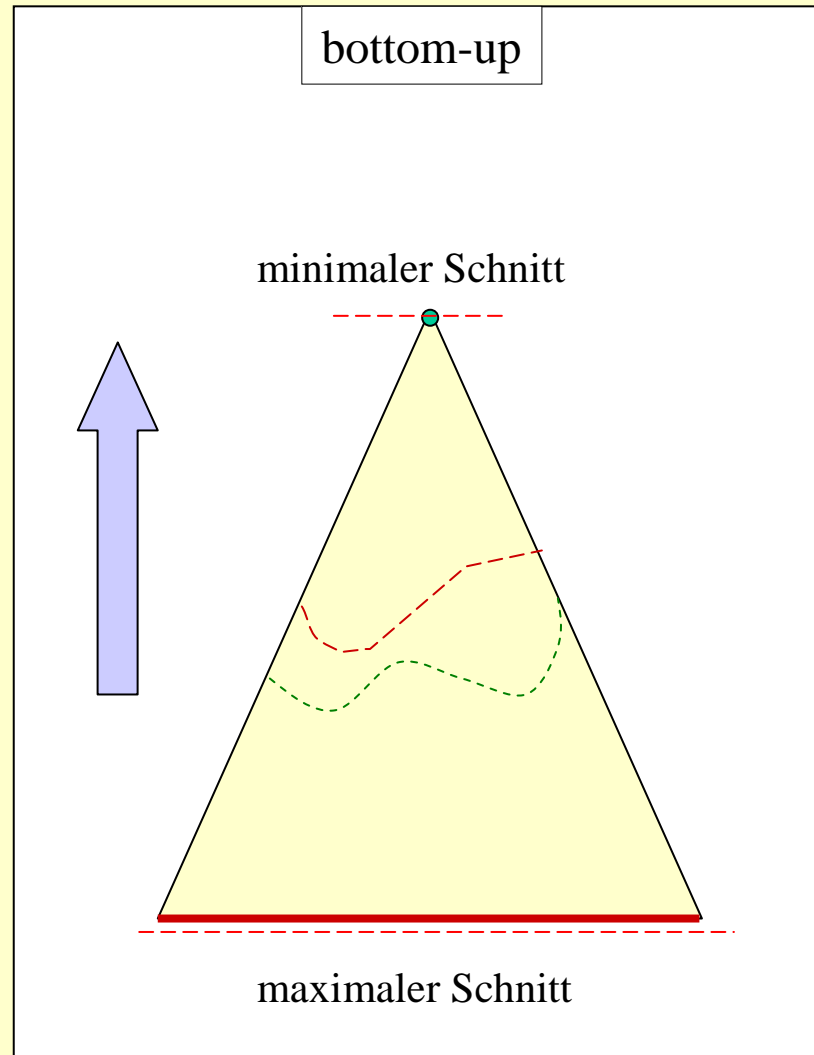
- oberstes Kellersymbol  $x = \alpha \in N$  und  $\delta(\alpha, t) = \emptyset$ :
  - Eingabe weiterrücken bis:  $t' \in \text{Follow}(\alpha)$  gefunden wird.  $\alpha$  vom Keller nehmen
  - oder:  $\alpha$  auf dem Keller lassen und Eingabe weiter bis:  $t' \in \text{First}(\alpha)$  auftritt
  - falls  $\epsilon \in \text{First}(\alpha)$ :  $\alpha$  vom Keller nehmen
- oberstes Kellersymbol  $x = t' \in T$  und  $t' \neq t$ 
  - entferne  $t'$  vom Keller, Eingabe um ein Zeichen nach rechts

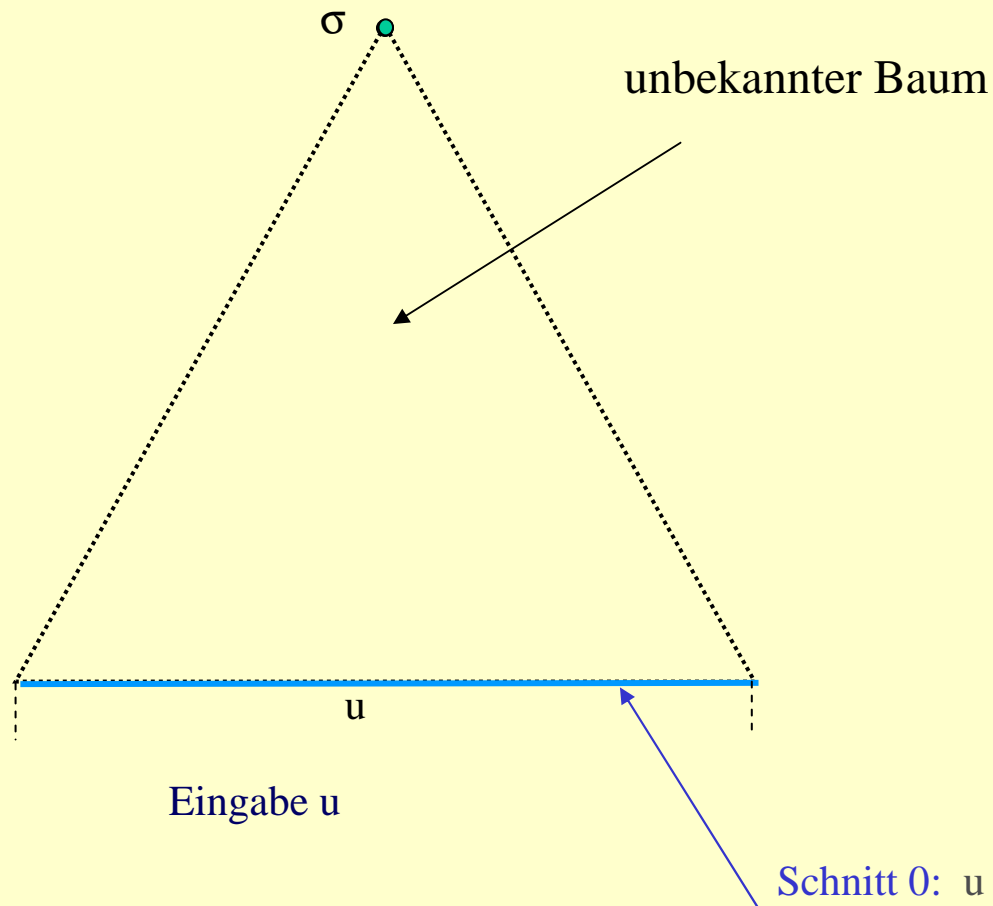


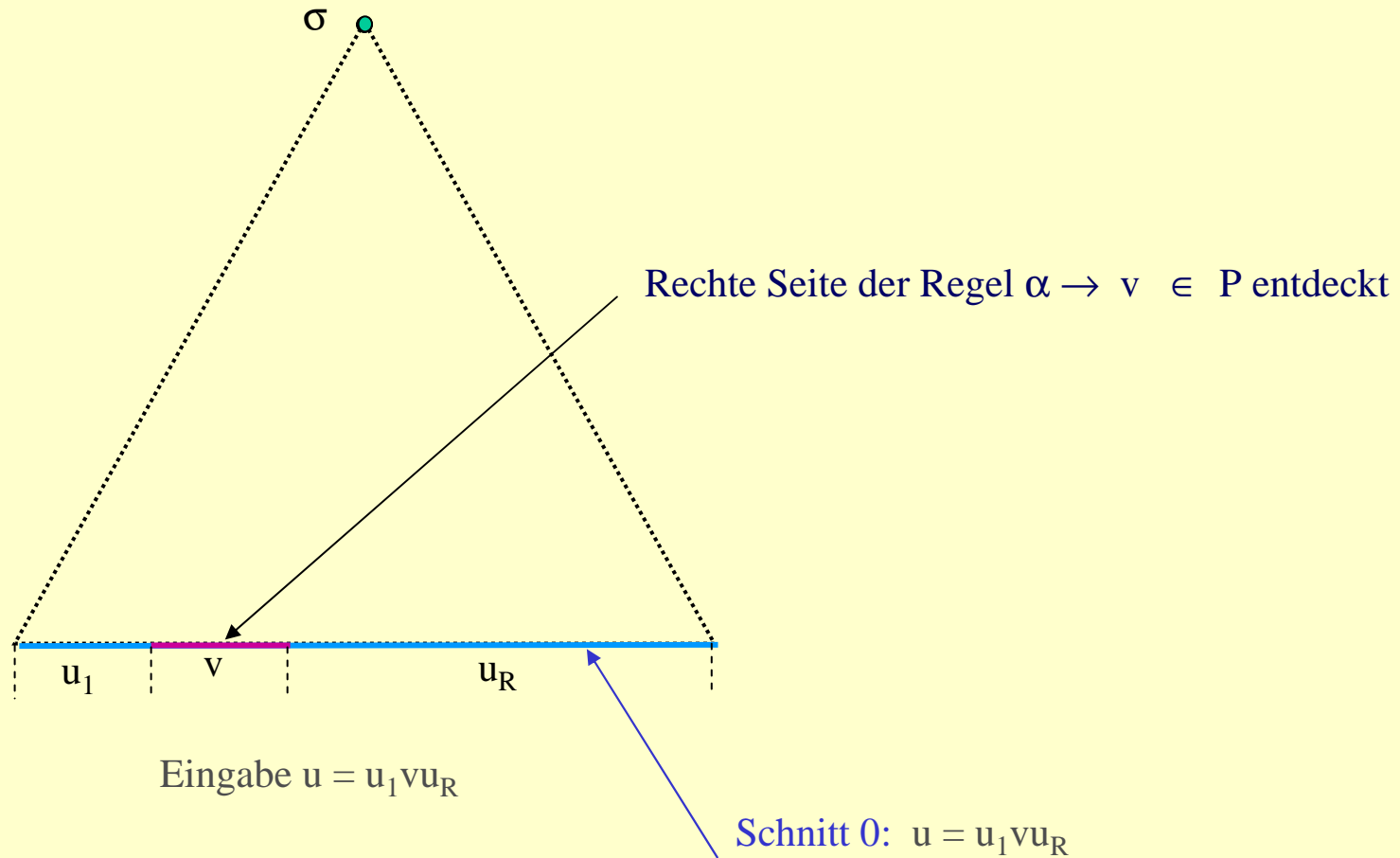
- oberstes Kellersymbol  $x = \alpha \in N$  und  $\delta(\alpha, t) = \emptyset$ :
  - Eingabe weiterrücken bis:  $t' \in \text{Follow}(\alpha)$  gefunden wird.  $\alpha$  vom Keller nehmen
  - oder:  $\alpha$  auf dem Keller lassen und Eingabe weiter bis:  $t' \in \text{First}(\alpha)$  auftritt
  - falls  $\epsilon \in \text{First}(\alpha)$ :  $\alpha$  vom Keller nehmen

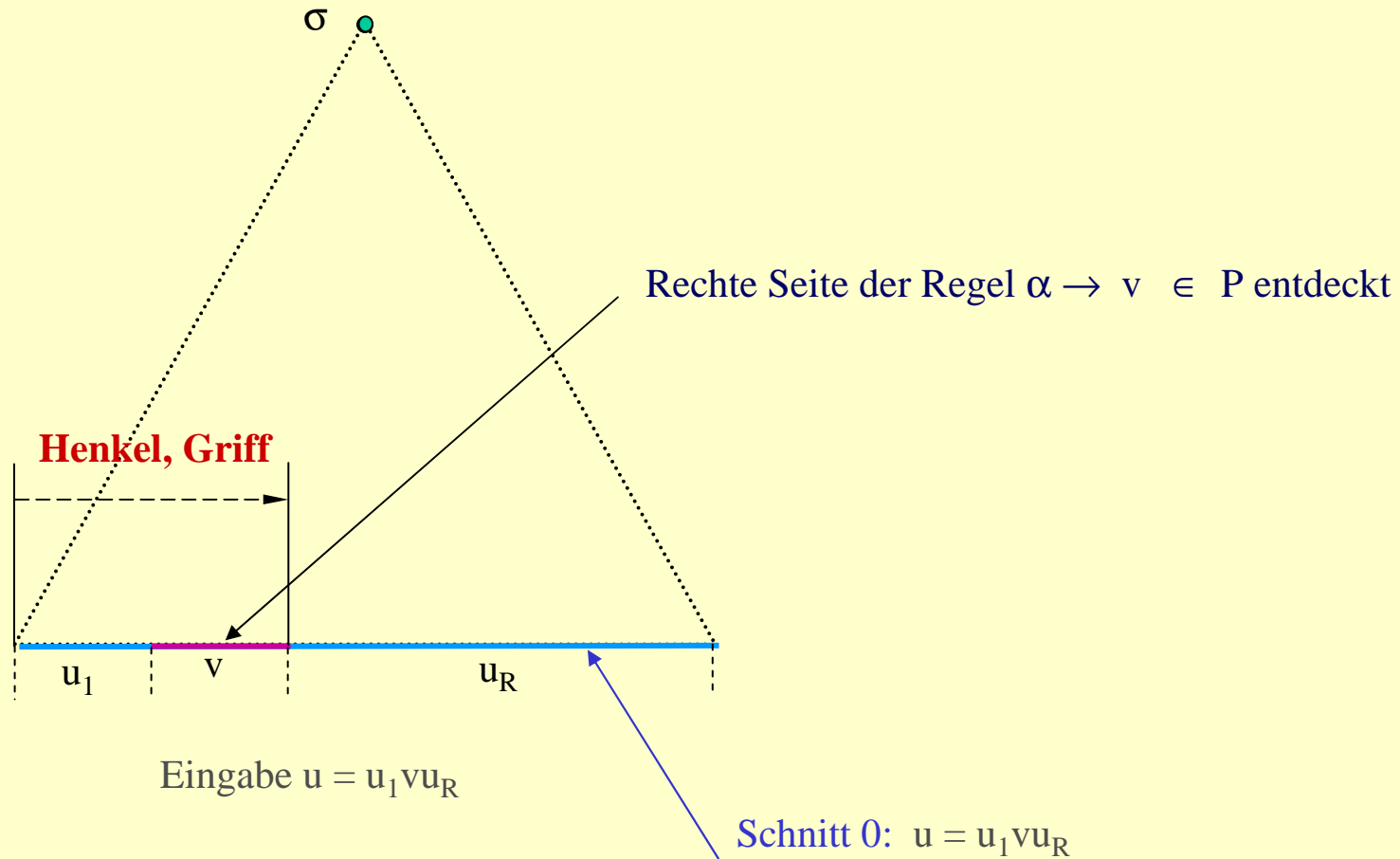


- oberstes Kellersymbol  $x = \alpha \in N$  und  $\delta(\alpha, t) = \emptyset$ :
  - Eingabe weiterrücken bis:  $t' \in \text{Follow}(\alpha)$  gefunden wird.  $\alpha$  vom Keller nehmen
  - **oder:  $\alpha$  auf dem Keller lassen und Eingabe weiter bis:  $t' \in \text{First}(\alpha)$  auftritt**
  - falls  $\epsilon \in \text{First}(\alpha)$ :  $\alpha$  vom Keller nehmen

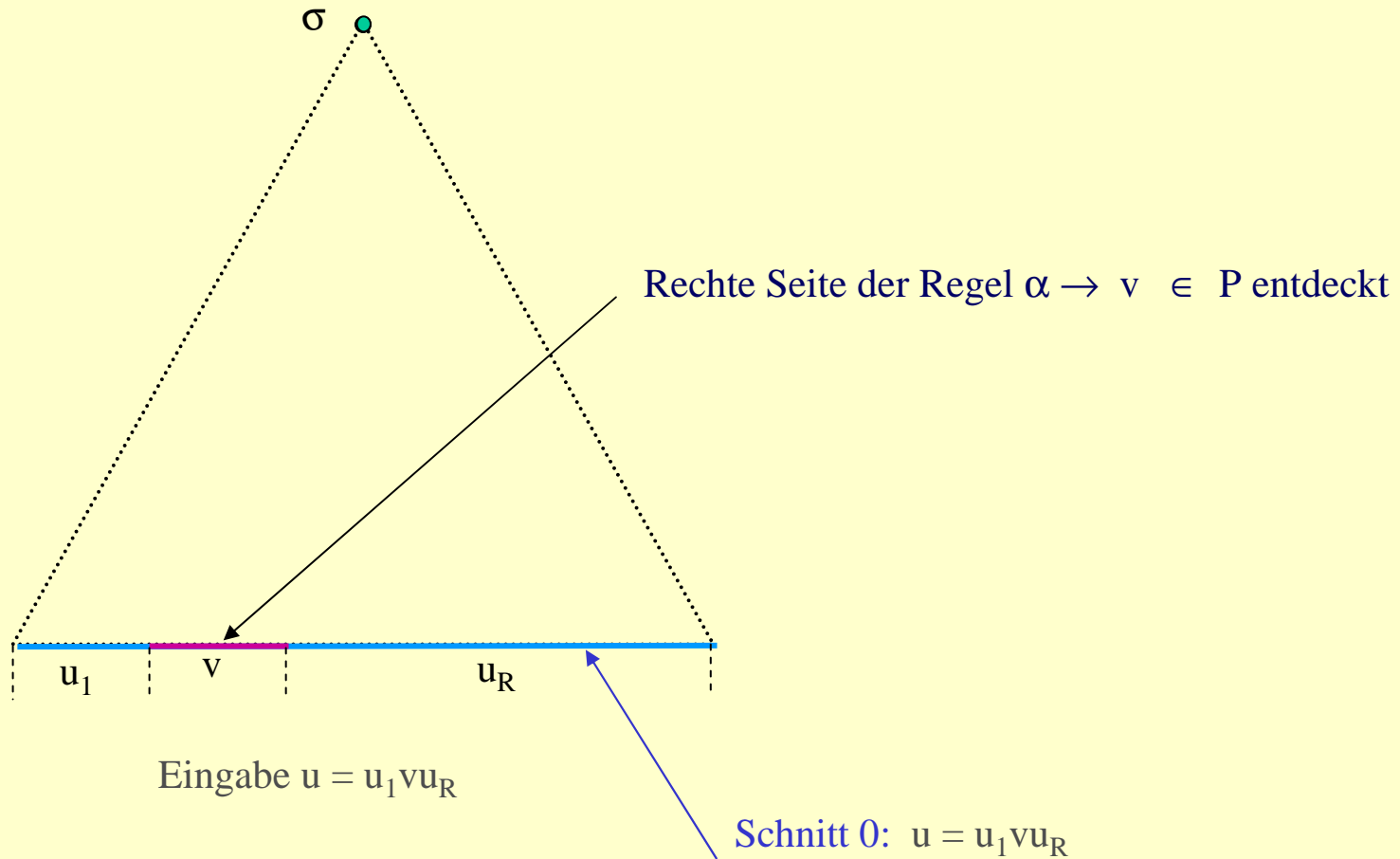




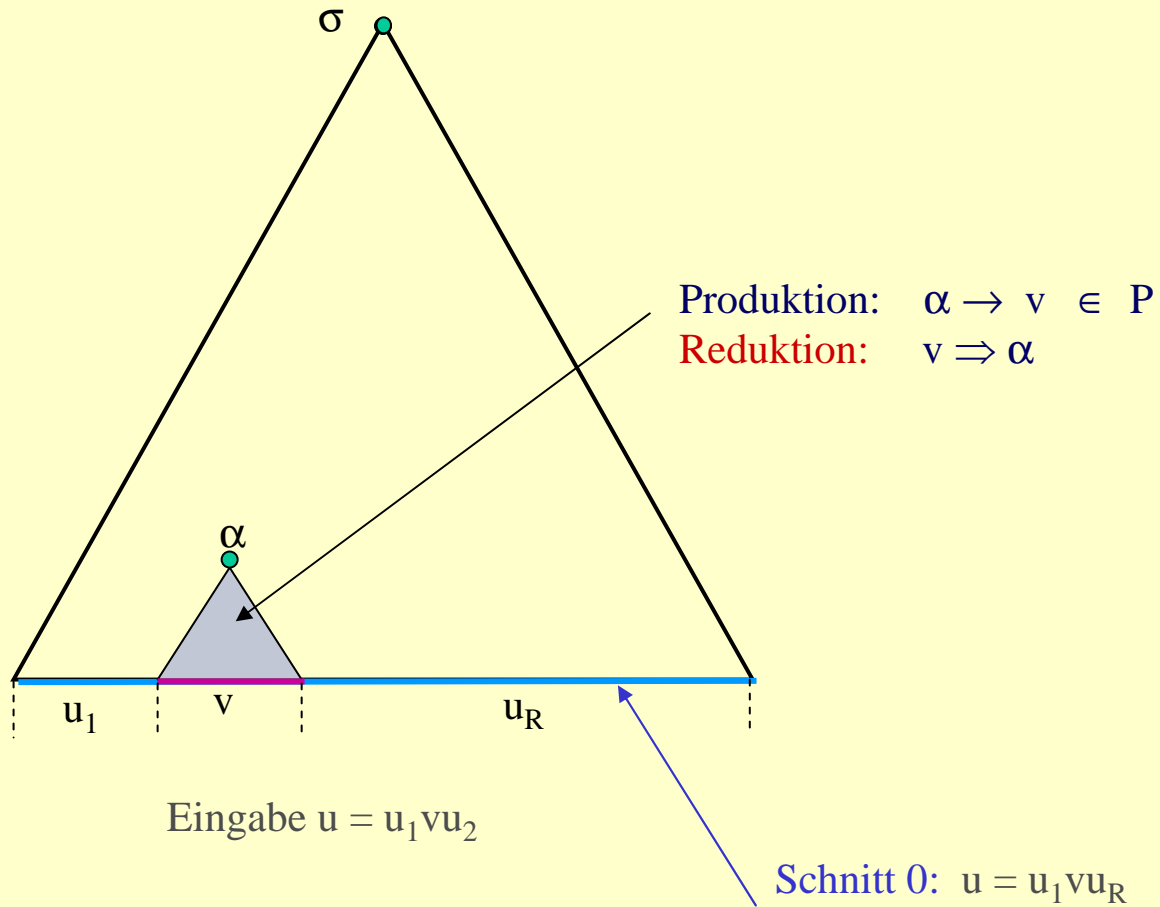




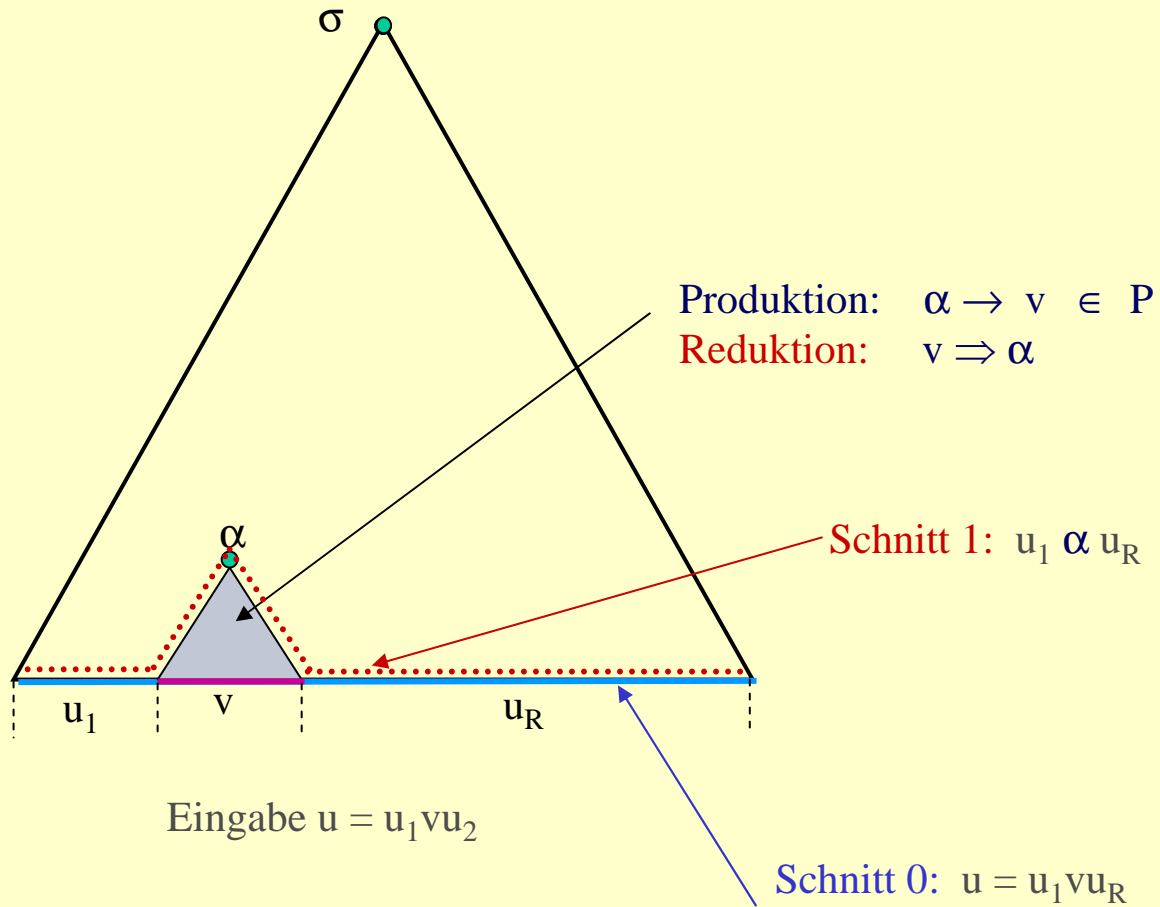
Ein *Henkel* ist ein Anfangsstück, das mit der rechten Seite einer Regel endet.



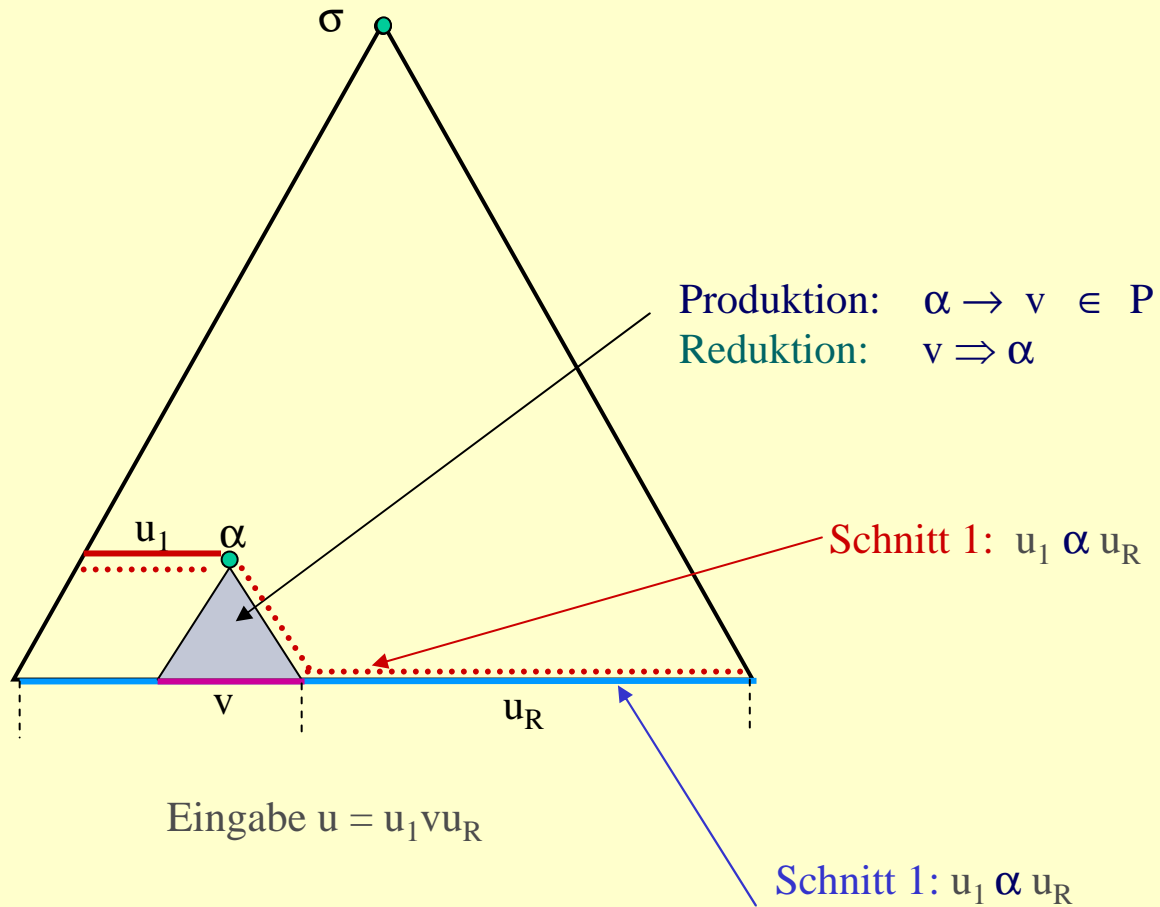
# Bottom-up Analyse

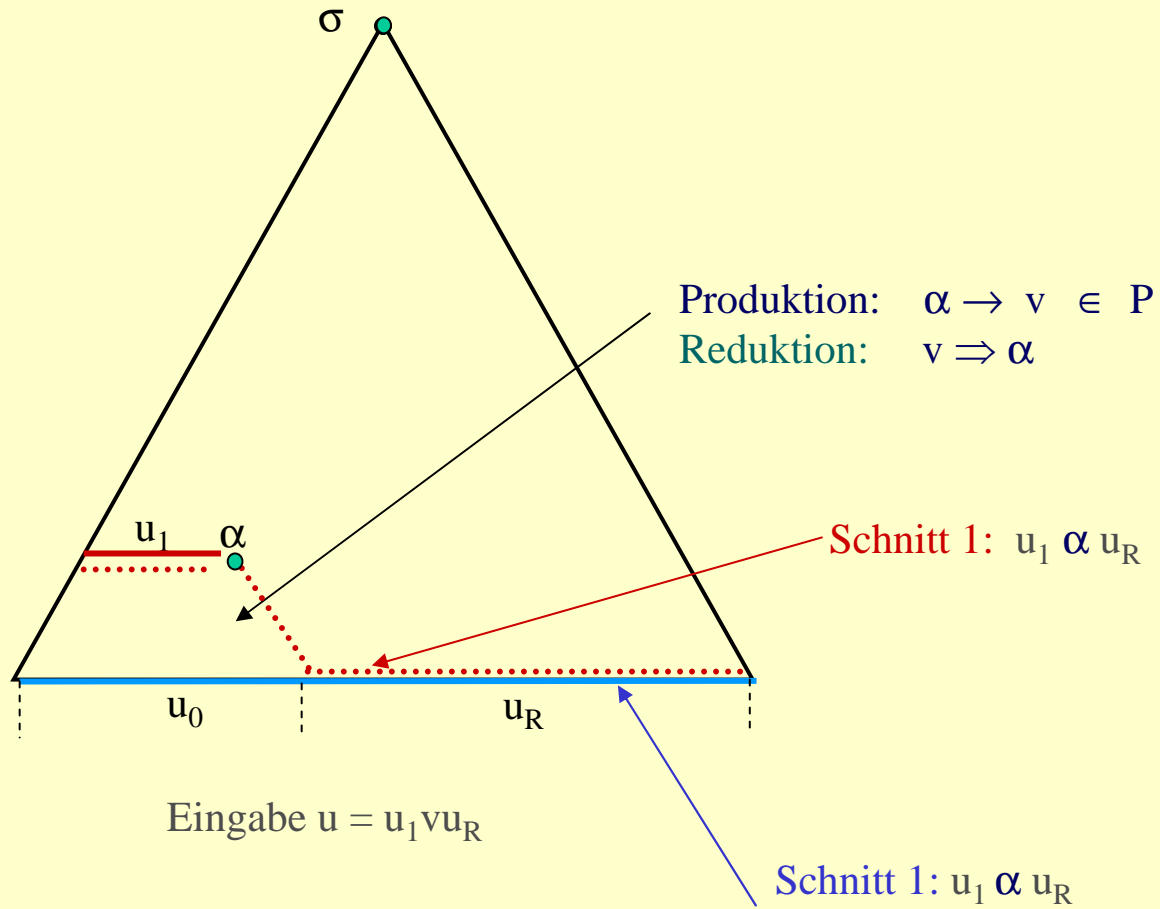


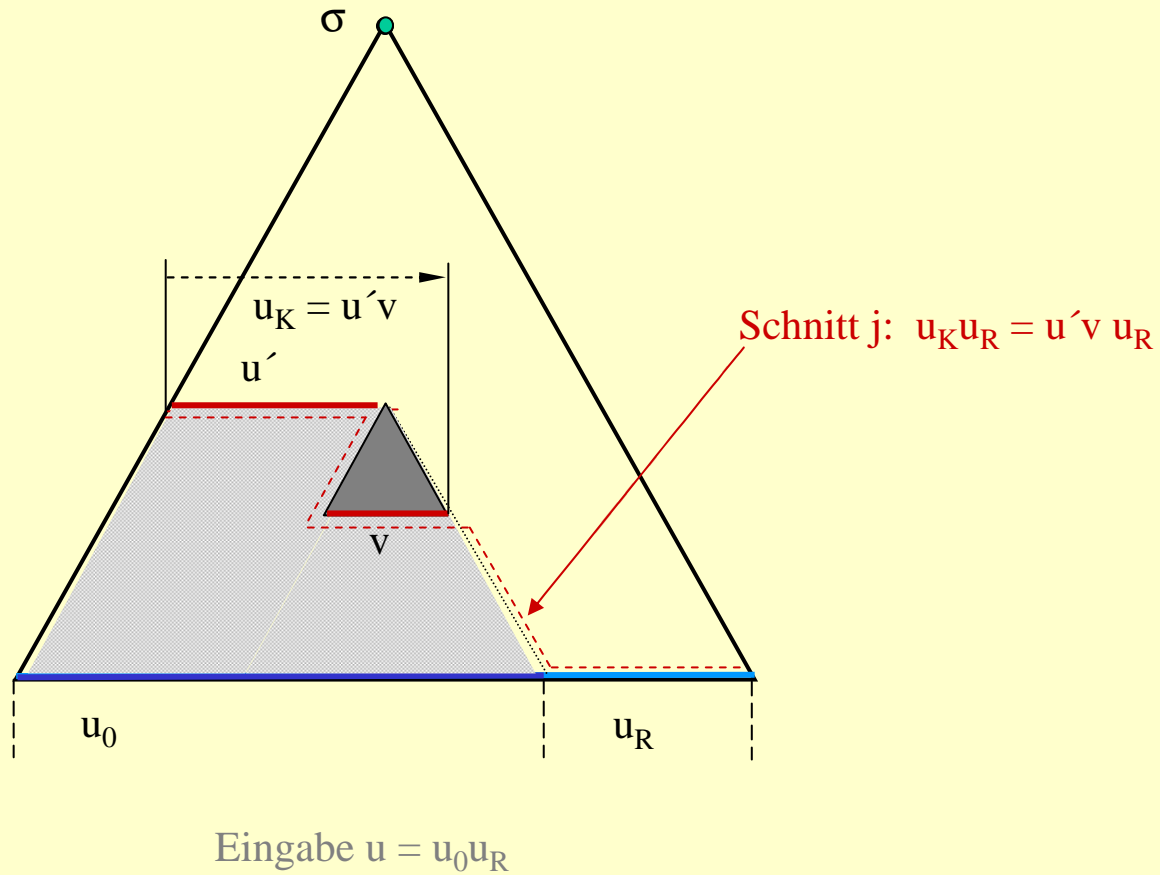
# Bottom-up Analyse

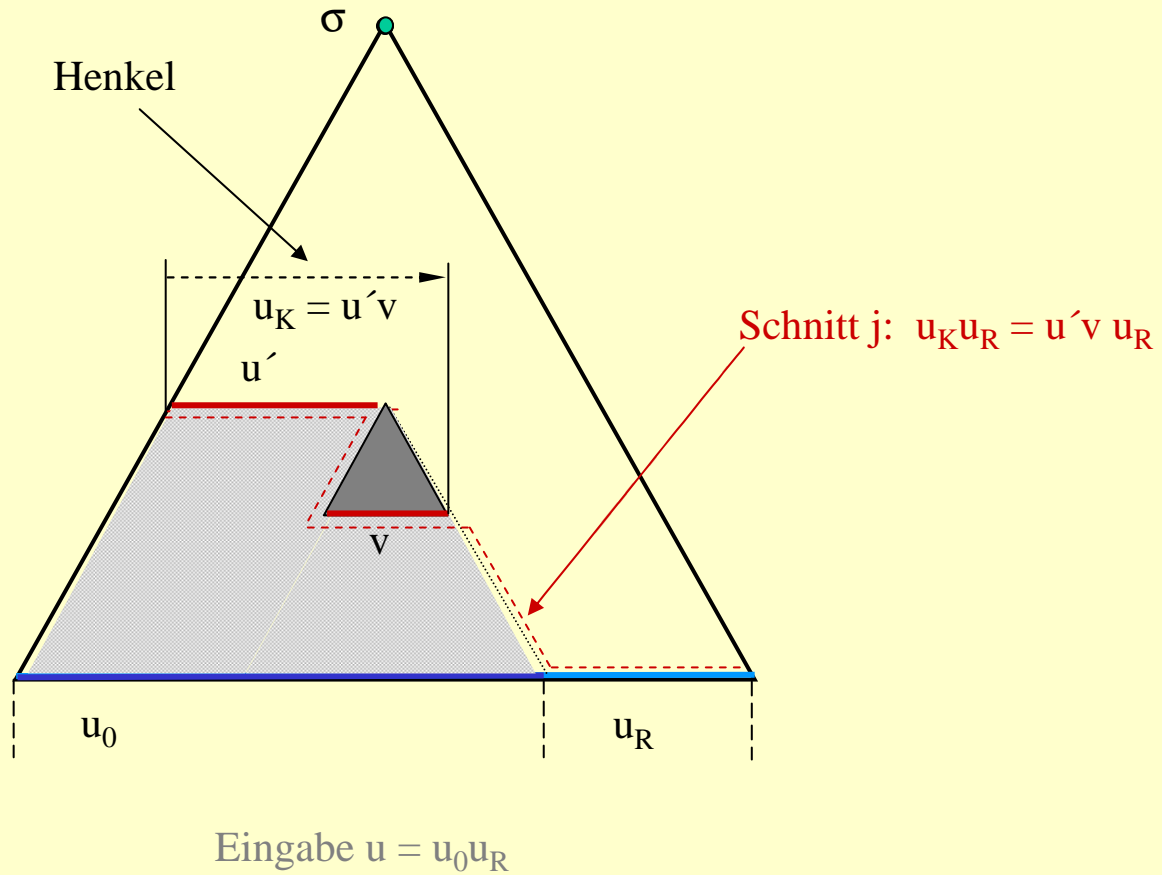


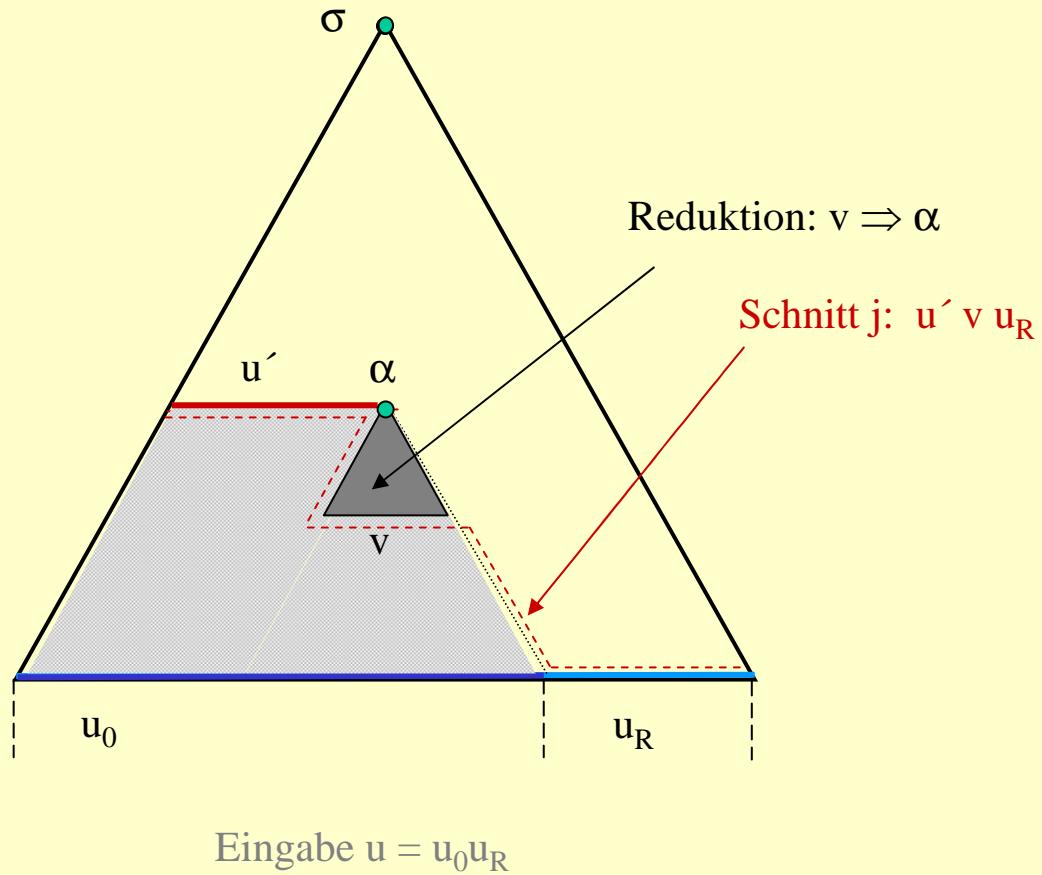
# Bottom-up Analyse

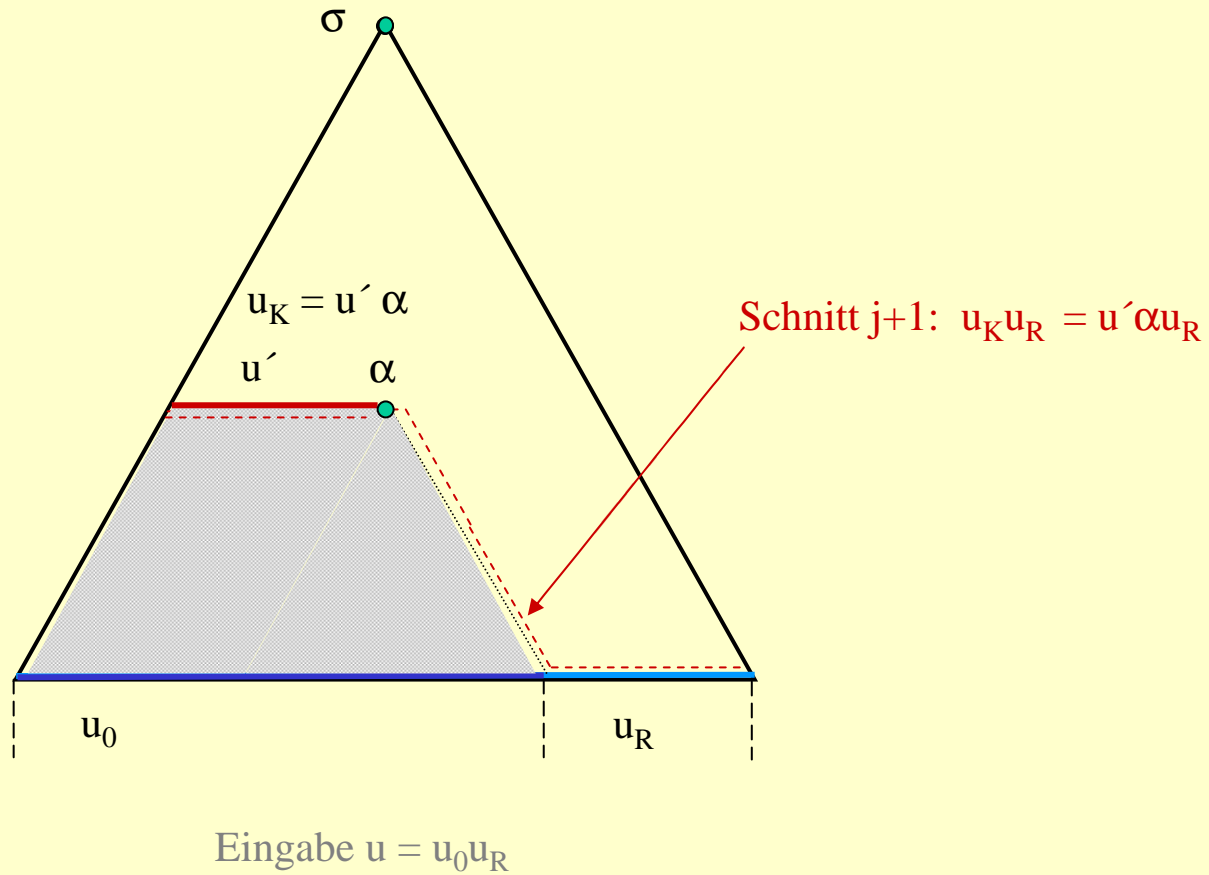


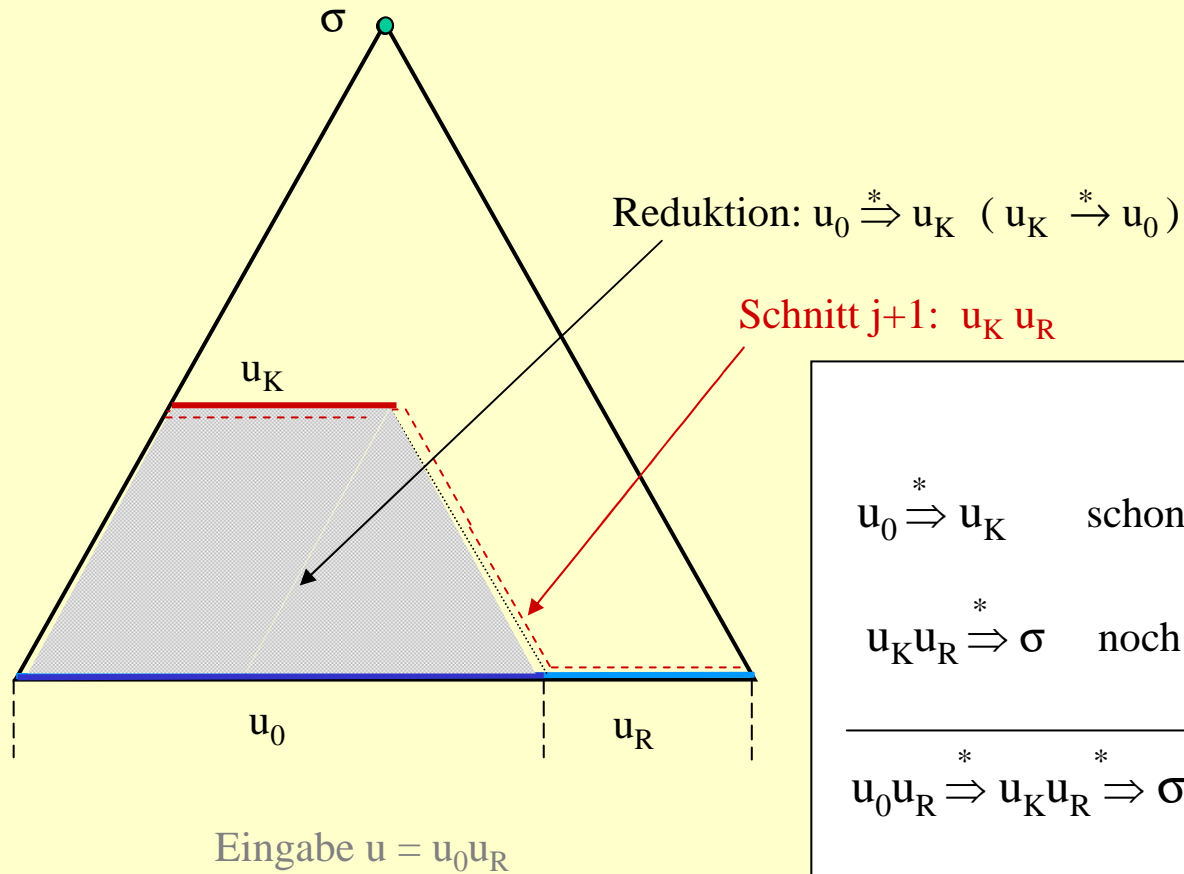










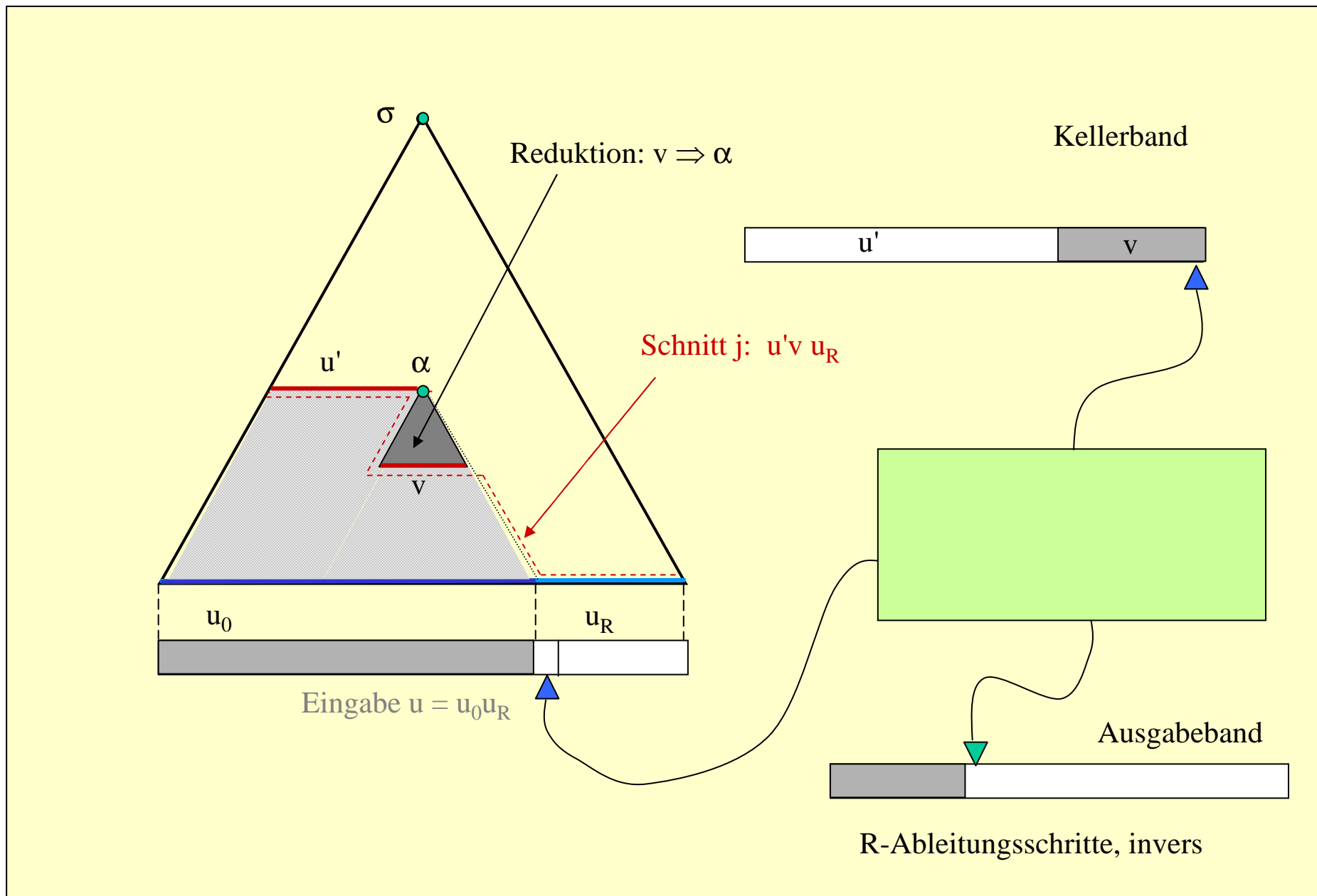


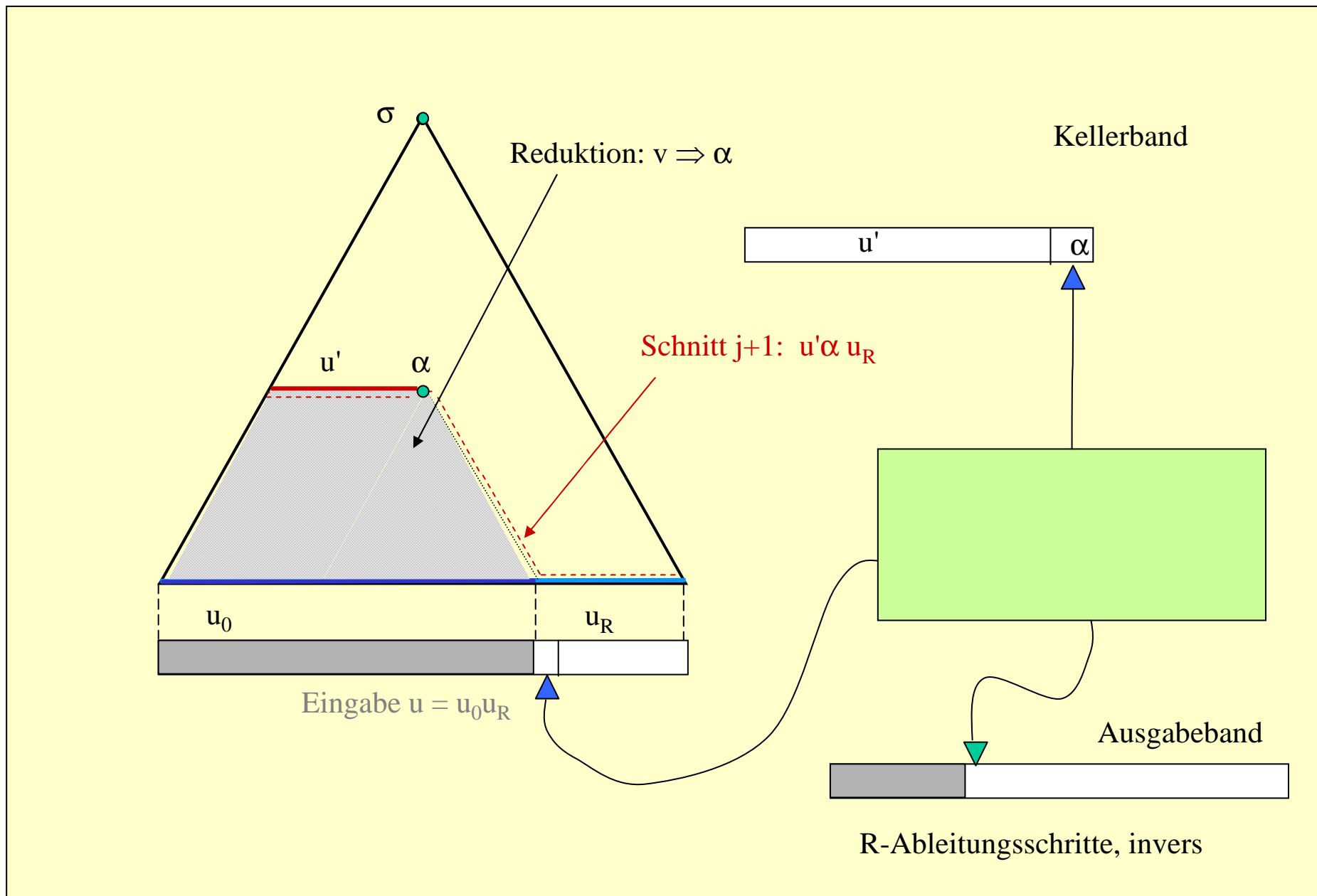
$u_0 \xRightarrow{*} u_K$  schon bewiesen  
 $u_K u_R \xRightarrow{*} \sigma$  noch zu beweisen  


---

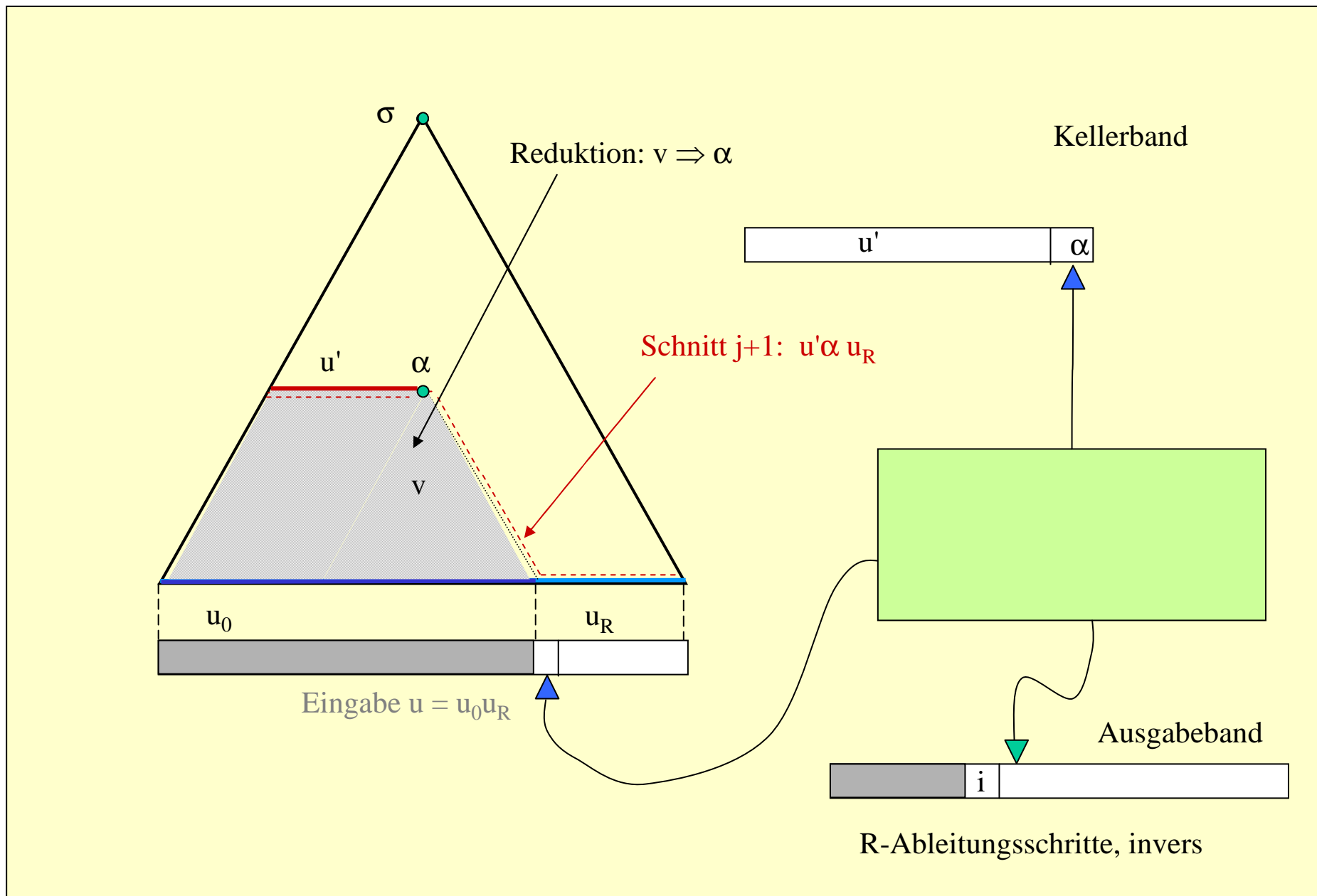
 $u_0 u_R \xRightarrow{*} u_K u_R \xRightarrow{*} \sigma$

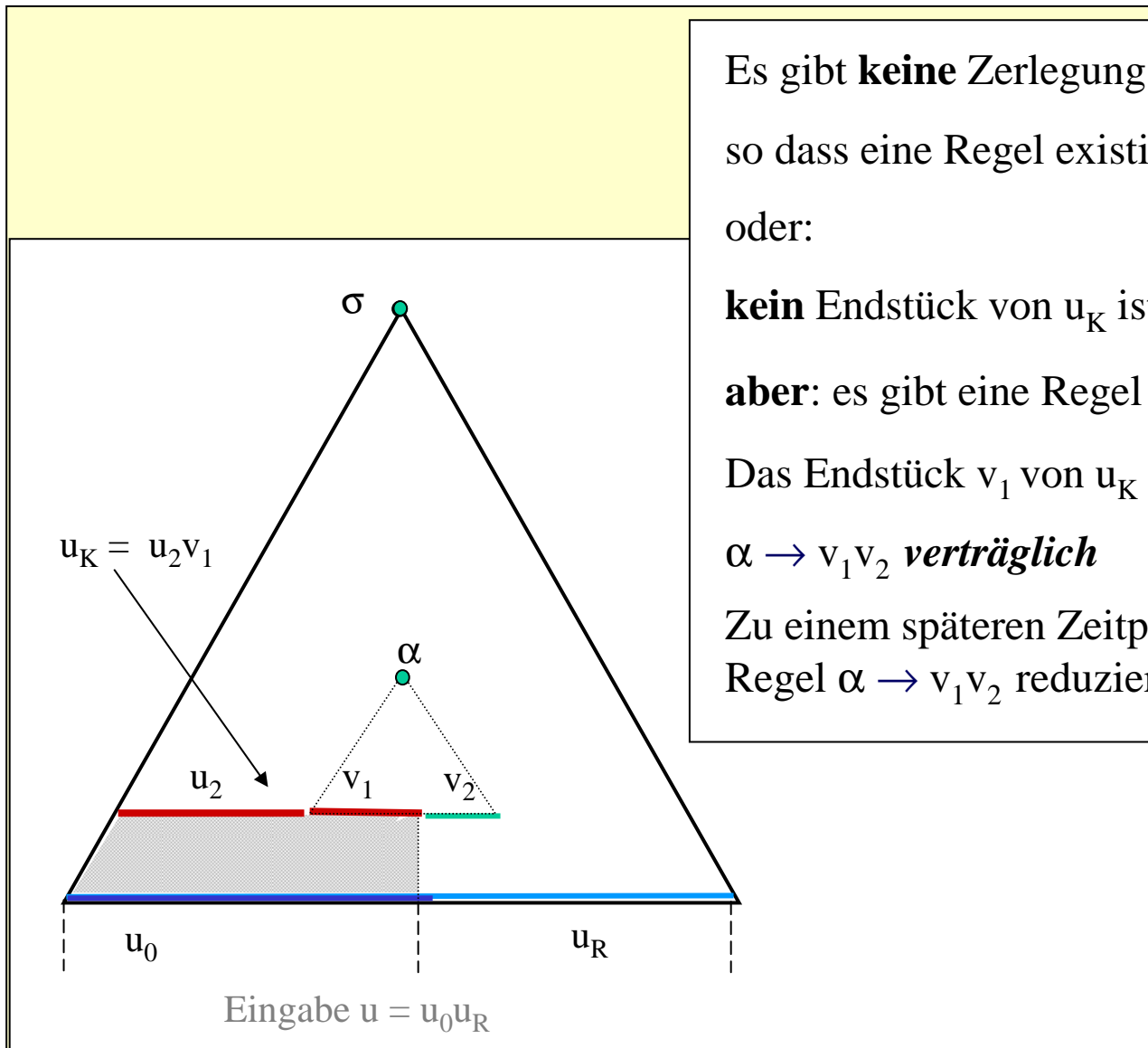
# Bottom-up Analyse mit Kellerautomat





# Bottom-up Analyse mit Kellerautomat





Es gibt **keine** Zerlegung  $u_K = u_2 v_1$

so dass eine Regel existiert der Form:  $\beta \rightarrow v_1$

oder:

**kein** Endstück von  $u_K$  ist **rechte Seite** einer Regel

**aber:** es gibt eine Regel  $\alpha \rightarrow v_1 v_2$

Das Endstück  $v_1$  von  $u_K$  ist mit der Regel

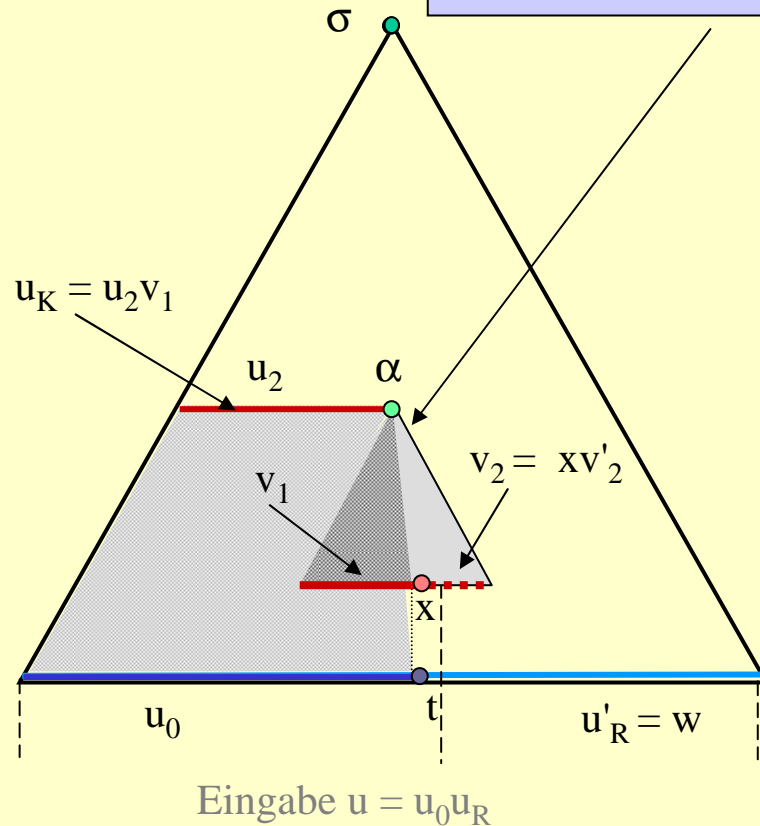
$\alpha \rightarrow v_1 v_2$  **verträglich**

Zu einem späteren Zeitpunkt könnte nach der Regel  $\alpha \rightarrow v_1 v_2$  reduziert werden



$\alpha \rightarrow v \in P$  und  $v = v_1 v_2$

aber Reduktion  $v \Rightarrow \alpha$  eventuell noch nicht möglich



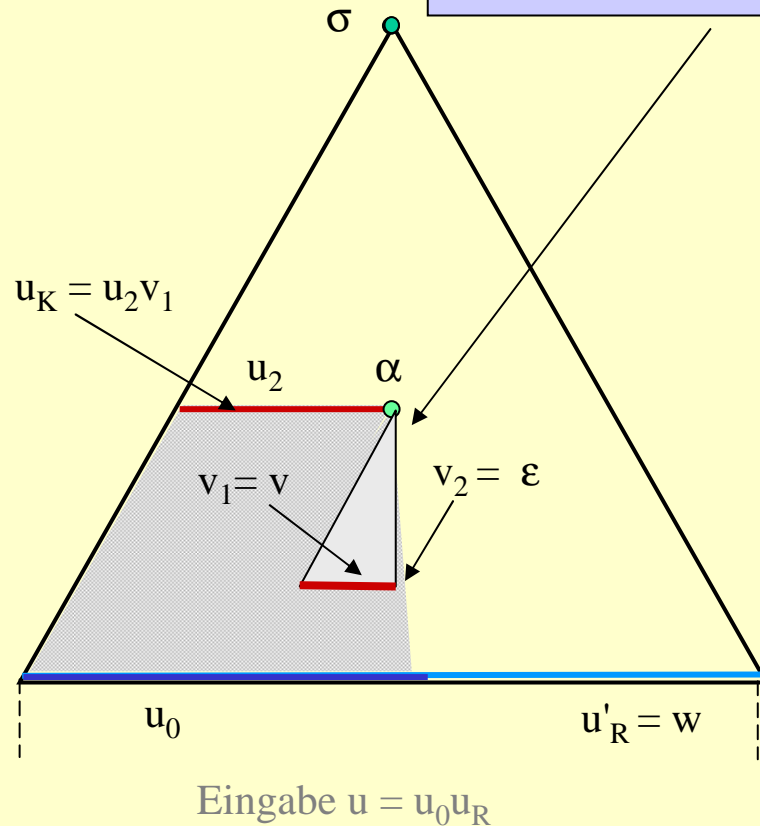
•  $x = t \in T$ :

$v_2 = t v'_2$ ,  $u_R = t w$

**shift:**  $u'_R = w$  und  $u'_K = u_K t$

$\alpha \rightarrow v \in P$  und  $v = v_1 v_2$

aber Reduktion  $v \Rightarrow \alpha$  eventuell noch nicht möglich

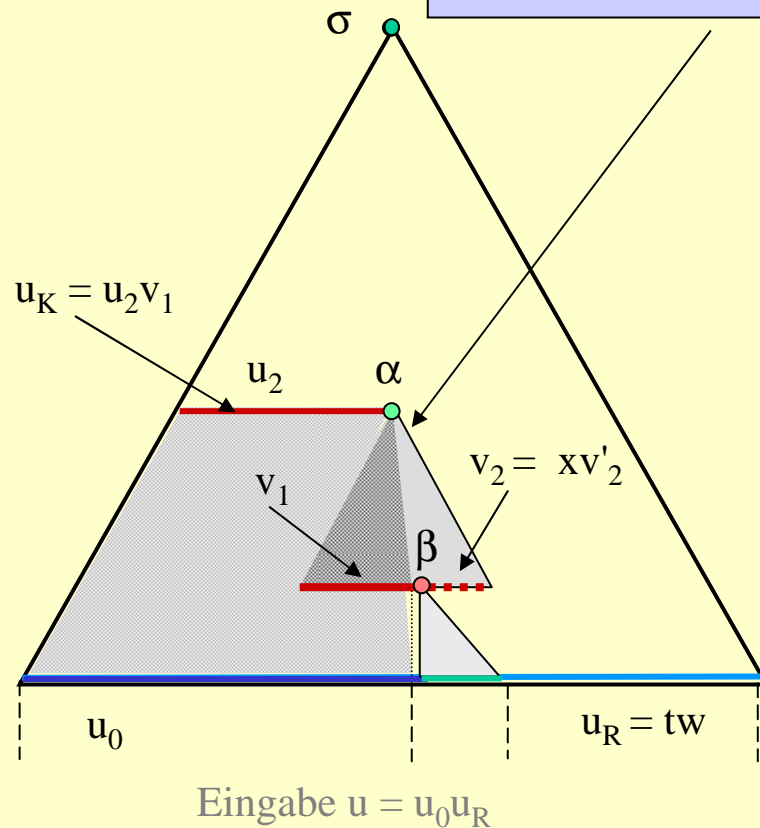


- $v_2 = \epsilon$  ( $v = v_1$ ):  
**reduce:** ersetze  $v_1$  auf dem Keller durch:  $\alpha$



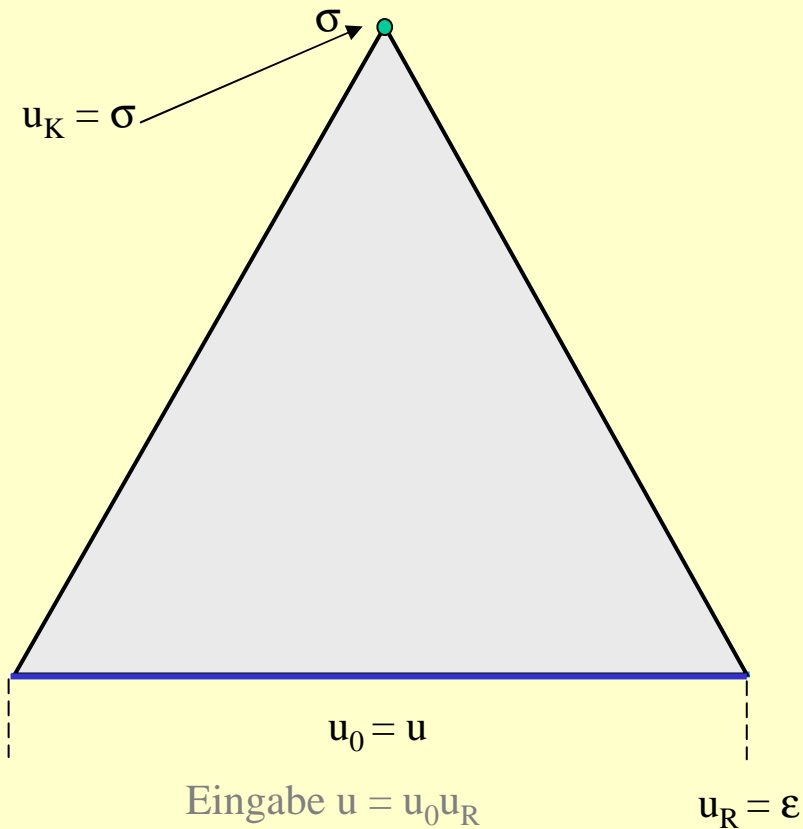
$\alpha \rightarrow v \in P$  und  $v = v_1 v_2$

aber Reduktion  $v \Rightarrow \alpha$  eventuell noch nicht möglich



•  $x = \beta \in N$ :  $v_2 = \beta v'_2$

**versuche** Anfangsstück von  $u_R$  auf  $\beta$  zu **reduzieren**

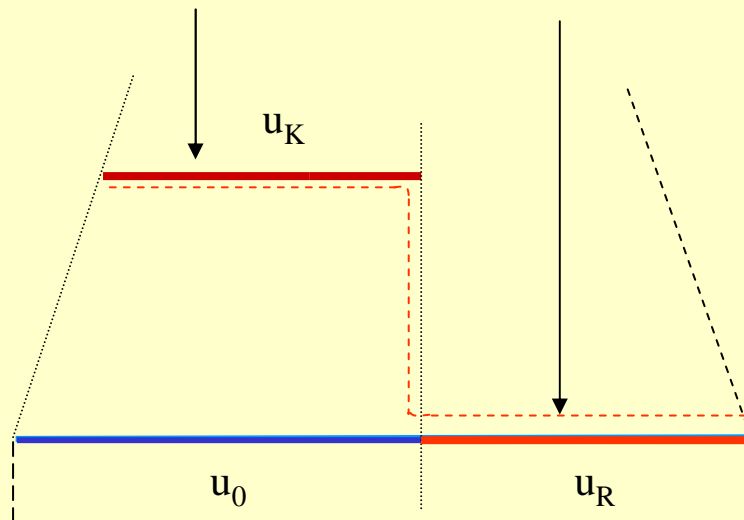


- $u_K = \sigma$  und  $u_R = \epsilon$

**accept:** akzeptiere die Eingabe



Der *Schnitt* wird durch die Zerlegung in das *Kellerwort*  $u_K$  und das *Restwort*  $u_R$  beschrieben.



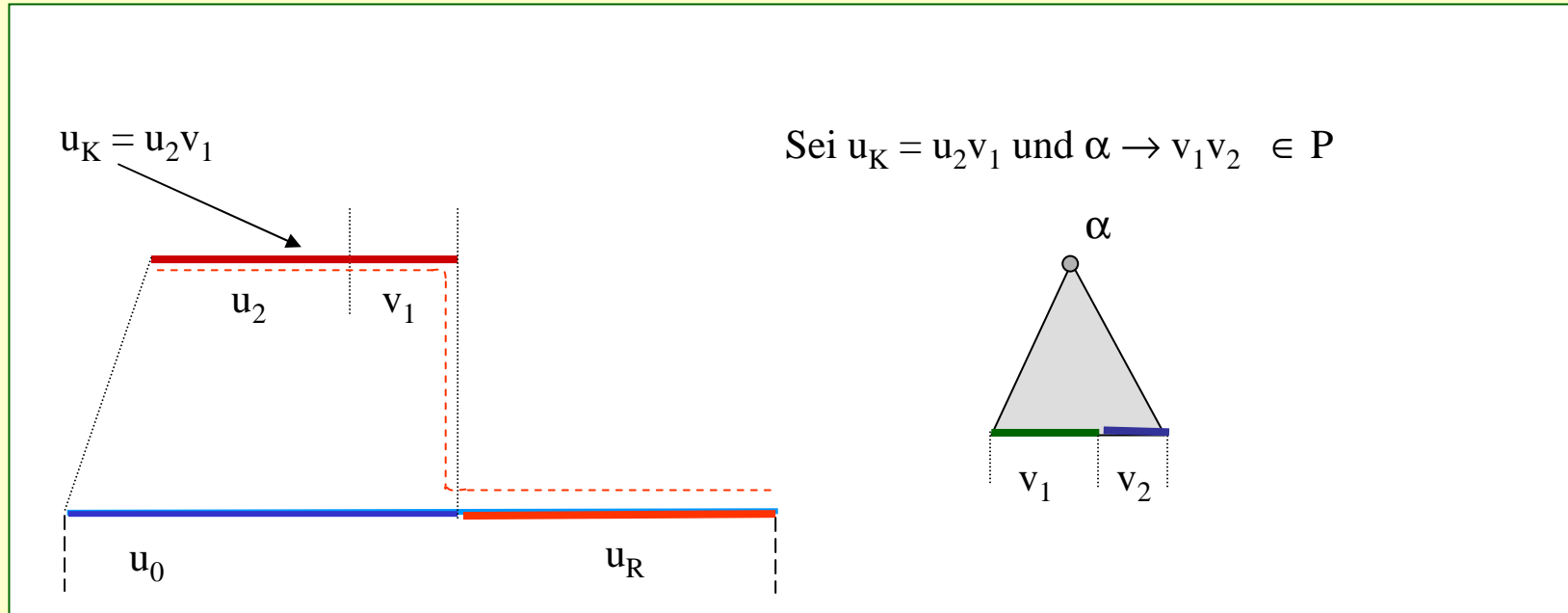
Ein Wort-Paar  $(u, w) \in V^* \times T^*$  heißt *Situation* und wird durch  $u \bullet w$  notiert.

Jeder solche Schnitt im Ableitungsbaum definiert eine spezielle *gültige Situation*  $u_K \bullet u_R$  für  $G$ , wobei zusätzlich gilt:

$$\sigma \xrightarrow[R]{*} u_K u_R \in S(G)$$

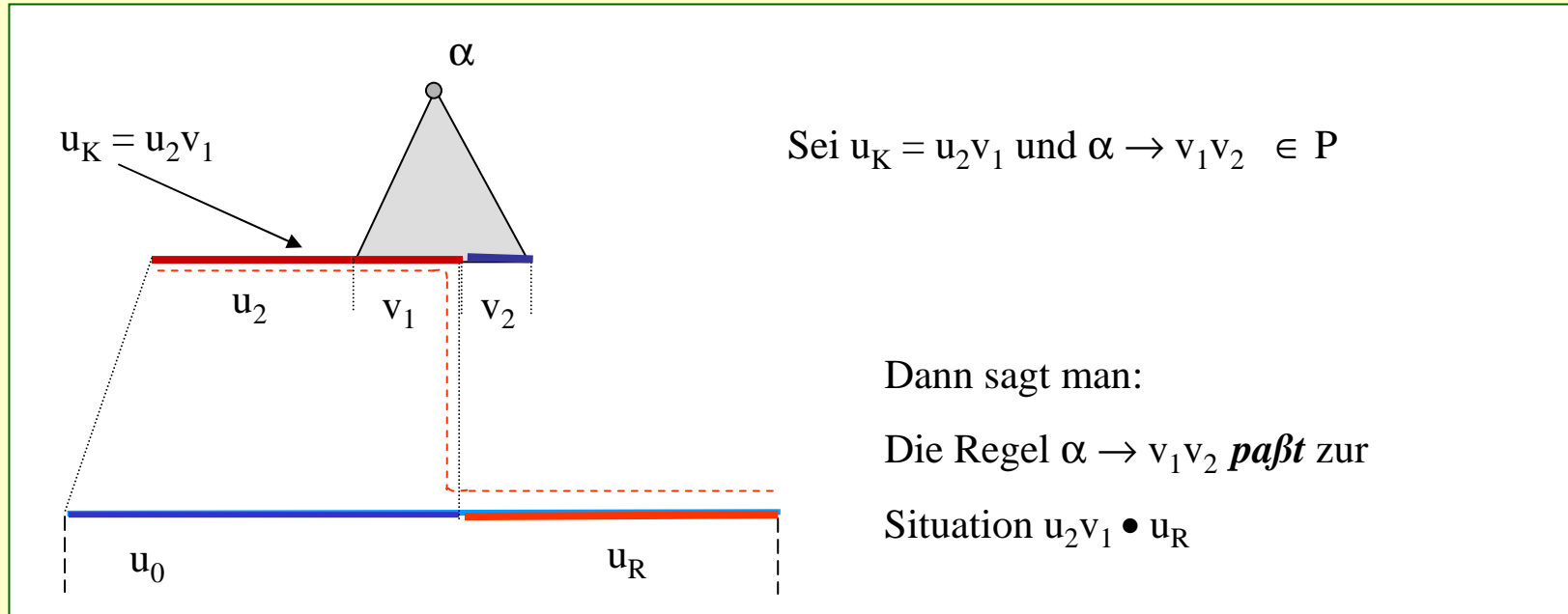
Sei  $\text{Sit}(G)$  die Menge der *gültigen Situationen* von  $G$  und

$\text{Pr}(G) = \{ u \in V^* \mid u \bullet w \in \text{Sit}(G) \}$  die Menge der *gültigen Präfixe* von  $G$





## Situation und punktierte Regel



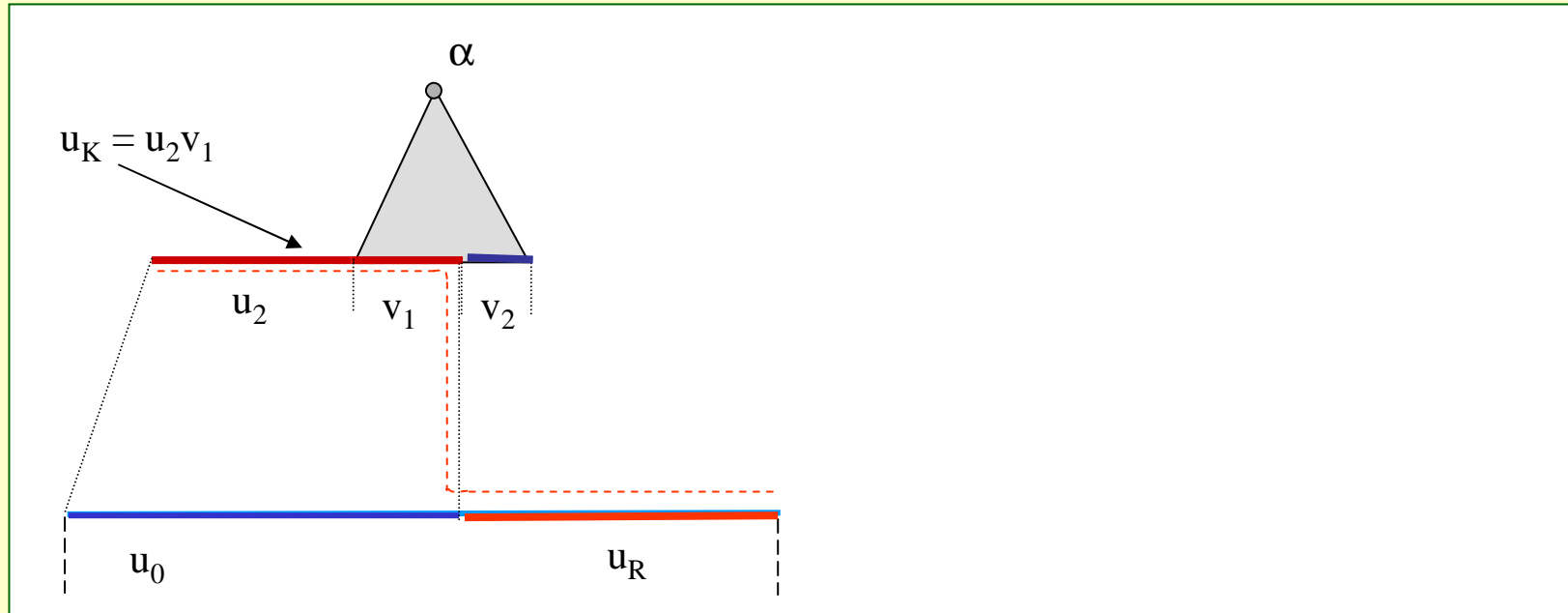
Für jede Regel  $\alpha \rightarrow v \in P$  führt man daher die *punktierten Regeln*

$$\alpha \rightarrow v_1 \bullet v_2$$

für jede Zerlegung  $v = v_1 v_2$  ein (auch mit  $v_1 = \varepsilon$  oder  $v_2 = \varepsilon$ )

Eine punktierte Regel heißt auch *LR(0)-Konfiguration* oder *cf item*

## Punktierte Regel als Hypothese



In der *Situation*  $u_2 v_1 \bullet u_R$  ist die folgende *Hypothese* zulässig:

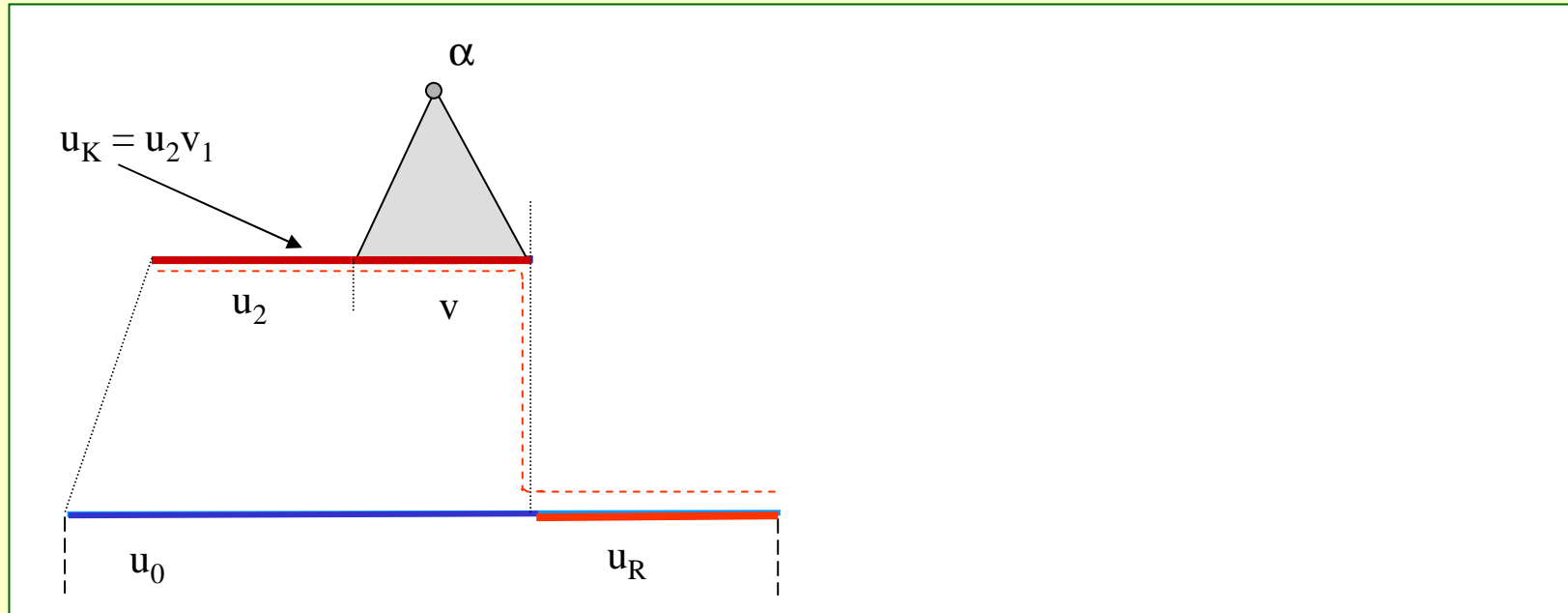
**Die Regel  $\alpha \rightarrow v = v_1 v_2$  kann später eventuell zur Reduktion verwendet werden**

Die punktierte Regel  $\alpha \rightarrow v_1 \bullet v_2$  ist mit dem Präfix  $u_2 v_1$  *verträglich*

Es wird versucht, zu einem Präfix die verträglichen Regeln zu finden:

**Pattern matching Problem**

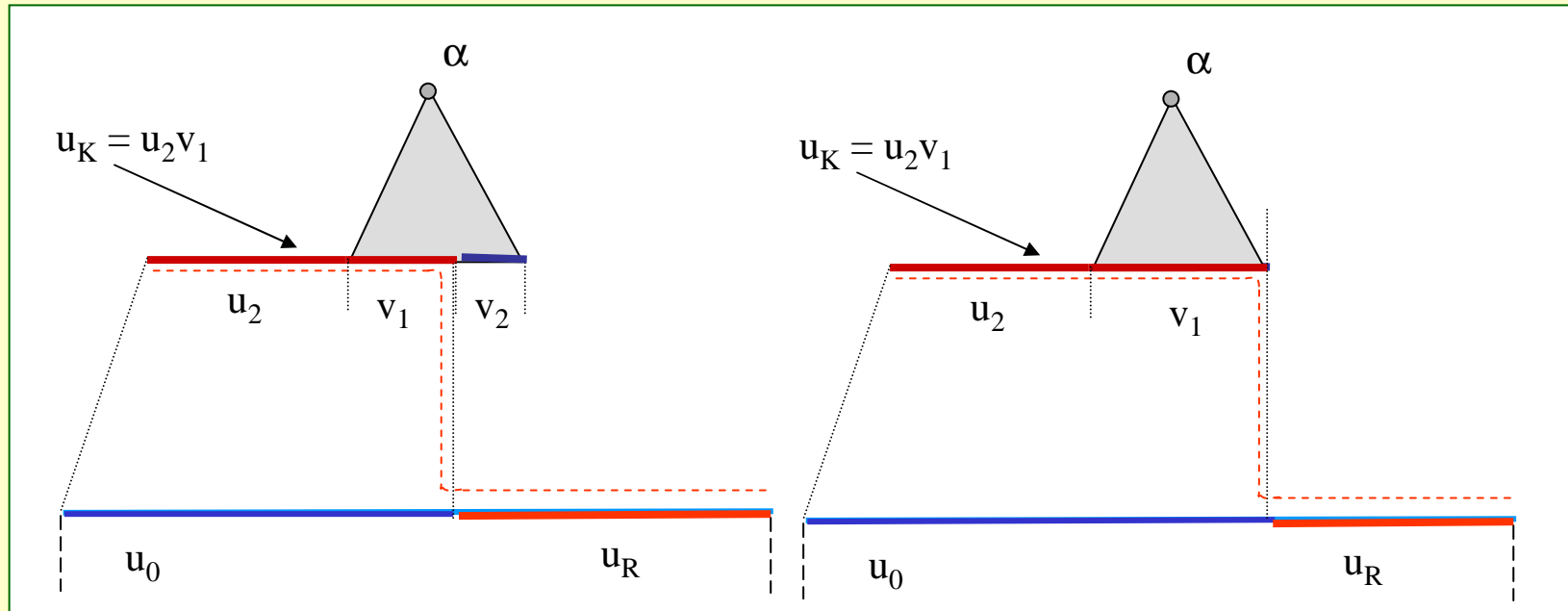
# Punktierte Regel als Hypothese



Gegeben sei die *Situation*  $u_2v \bullet u_R$  mit dem Präfix  $u_2v$  und es existiere eine *Regel*  $\alpha \rightarrow v \in P$ , dann ist  $\alpha \rightarrow v \bullet$  verträglich mit  $u_2v$  und man kann nach der Regel  $\alpha \rightarrow v$  *reduzieren*:  $u_2v \bullet u_R \Rightarrow u_2 \alpha \bullet u_R$

Eine solche Situation  $u_2v \bullet u_R$  heißt dann auch ein *Henkel* oder *Griff* der Satzform  $u_2vu_R$  (zur Regel  $\alpha \rightarrow v \in P$ )

# Partielle und vollständige Henkel



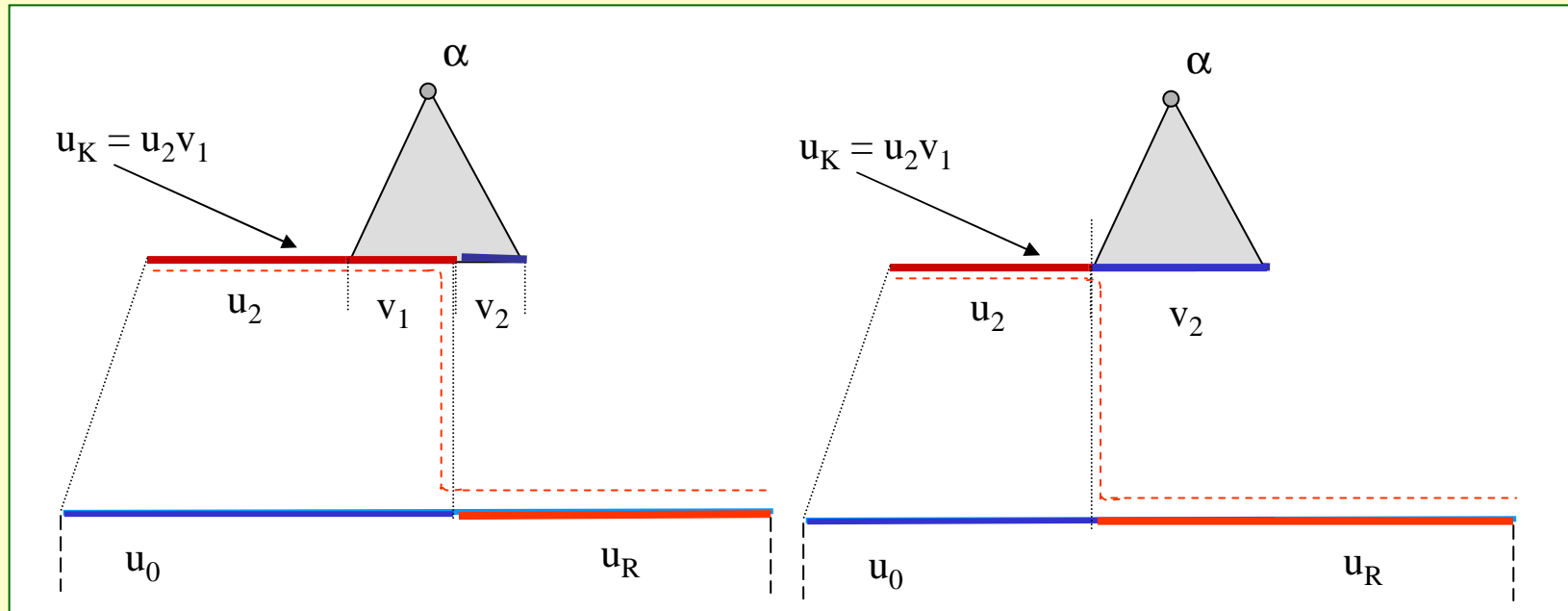
Eine *Situation*  $u_2 v_1 \bullet u_R$ , die mit der punktierten Regel  $\alpha \rightarrow v_1 \bullet v_2$  verträglich ist, heißt auch *partieller Henkel* für  $\alpha \rightarrow v_1 v_2$  (oder *partielles matching*)

Der Henkel heißt:

*vollständig*, wenn  $v_2 = \epsilon$  ist.

$\alpha \rightarrow v_1 \bullet$  heißt dann auch *akzeptierende Konfiguration*

# Partielle und vollständige Henkel



Eine *Situation*  $u_2 v_1 \bullet u_R$ , die mit der punktierten Regel  $\alpha \rightarrow v_1 \bullet v_2$  verträglich ist, heißt auch *partieller Henkel* für  $\alpha \rightarrow v_1 v_2$  (oder *partielles matching*)

Der Henkel heißt:

*total unvollständig* (oder *triviales matching*), wenn  $v_1 = \varepsilon$

Jeder Präfix ist verträglich mit einer punktierten Regel  $\alpha \rightarrow \bullet v_2$