Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Sprachverarbeitung II / 4 FS 2017

Two-Level-Regeln und Transduktoren

Buch: Kapitel 6.6

Beat Pfister

Sprachverarbeitung II / 4

Vorlesung: **Sprachsynthese** (Teil II.4)

Morphologische Analyse (Wortanalyse):

- Two-Level-Regeln
- Transduktoren

Übung: Entwicklung einer DCG-Satzgrammatik

Grammatik für natürlichsprachliche Sätze

DCG-Formalismus (*definite clause grammar*)

- Eignet sich zur Beschreibung natürlicher Sprachen
  (Verschachtelung erfordert mindestens Typ-2-Grammatik)
- Übereinstimmungsphänomene (z.B. KNG in NG) sind mit Attributen beschreibbar (Unifikation erzwingt Übereinstimmung)
- Dank Attributen effizient $\longrightarrow$ geringe Anzahl Regeln nötig
- kontextfreies Skelett erlaubt Chart-Parsing $\longrightarrow$ Syntaxbaum

Grammatik für natürlichsprachliche Sätze

Satzgrammatik mit Wörtern als Terminalsymbole ist unpraktisch, weil

die Zahl der Terminalsymbole sehr gross ist!

- Nomen, Verben und Adjektive haben Flexionsformen
  (Flexion = Deklination von Nomen und Adjektiven + Konjugation von Verben)

- praktisch unbegrenzte Kompositionsmöglichkeiten
  (Kompositum: zusammengesetztes Wort)

$\longrightarrow$ Wortgrammatik mit Morphemen als Terminalsymbole

Wortgrammatik

Beschreibt Aufbau von Wörtern aus Morphemen

“legen”	$\longrightarrow$	“leg” + “en”
“gelegen”	$\longrightarrow$	“ge” + “leg” + “en”
“gelegenheit”	$\longrightarrow$	“ge” + “leg” + “en” + “heit”
“angelegenheit”	$\longrightarrow$	“an” + “ge” + “leg” + “en” + “heit”
“angelegenheiten”	$\longrightarrow$	“an” + “ge” + “leg” + “en” + “heit” + “en”
“staatsangelegenheiten”	$\longrightarrow$	“staat” + “s” + “an” + “ge” + “leg” + “en” + “heit” + “en”

Wortgrammatik

Beschreibt Aufbau von Wörtern aus Morphemen

“legen”	$\longrightarrow$	“leg” + “en”
“gelegen”	$\longrightarrow$	“ge” + “leg” + “en”
“gelegenheit”	$\longrightarrow$	“ge” + “leg” + “en” + “heit”
“angelegenheit”	$\longrightarrow$	“an” + “ge” + “leg” + “en” + “heit”
“angelegenheiten”	$\longrightarrow$	“an” + “ge” + “leg” + “en” + “heit” + “en”
“staatsangelegenheiten”	$\longrightarrow$	“staat” + “s” + “an” + “ge” + “leg” + “en” + “heit” + “en”

Probleme:

- Wörter fast beliebig komplex
- Morphemgrenzen nicht markiert

Zerlegung von Wörtern in Morpheme

Problem 1: Zerlegung in Morpheme ist nicht eindeutig

“angelegenheit” $\longrightarrow$ “an” + “ge” + “leg” + “en” + “heit”

$\longrightarrow$ “angel” + “e” + “gen” + “heit”

$\Rightarrow$ morphologische Regeln (DCG): schränken Kombinierbarkeit der Morpheme ein

Zerlegung von Wörtern in Morpheme

Problem 1: Zerlegung in Morpheme ist nicht eindeutig

“angelegenheit” $\longrightarrow$ “an” + “ge” + “leg” + “en” + “heit”

$\longrightarrow$ “angel” + “e” + “gen” + “heit”

$\Rightarrow$ morphologische Regeln (DCG): schränken Kombinierbarkeit der Morpheme ein

Problem 2: beim Zusammenfügen veränderte Morpheme

“schiff” + “fahrt” $\longrightarrow$ “schiffahrt” oder “schifffahrt”

“be” + “handel” + “e” $\longrightarrow$ “behandle”

$\Rightarrow$ Lösung?

Zerlegung von Wörtern in Morpheme

Problem 1: Zerlegung in Morpheme ist nicht eindeutig

“angelegenheit” $\longrightarrow$ “an” + “ge” + “leg” + “en” + “heit”

$\longrightarrow$ “angel” + “e” + “gen” + “heit”

$\Rightarrow$ morphologische Regeln (DCG): schränken Kombinierbarkeit der Morpheme ein

Problem 2: beim Zusammenfügen veränderte Morpheme

“schiff” + “fahrt” $\longrightarrow$ “schiffahrt” oder “schifffahrt”

“be” + “handel” + “e” $\longrightarrow$ “behandle”

$\Rightarrow$ Two-Level-Regeln

Two-Level-Regeln

Zusammenhang zwischen lexikalischer Ebene und Oberflächenebene

lexikalische Ebene: `s c h i f f f a h r t`

(aneinandergefügte Morphe)

Oberflächenebene: `s c h i f ε f a h r t`

(korrekte Wortform)

Two-Level-Regel: $f{:}\varepsilon \Rightarrow f{:}f \_ f{:}f$

Two-Level-Regeln

Zusammenhang zwischen lexikalischer Ebene und Oberflächenebene

lexikalische Ebene: `s c h i f f f a h r t`

(aneinandergefügte Morphe)

Oberflächenebene: `s c h i f ε f a h r t`

(korrekte Wortform)

Two-Level-Regel: $f{:}\varepsilon \Rightarrow f{:}f \_ f{:}f$

Kopf

Operator

Bedingungsteil

Two-Level-Regeln: Notation und Bedeutung

1. $x{:}y \Rightarrow LC\_RC$
2. $x{:}y \Leftarrow LC\_RC$
3. $x{:}y \Leftrightarrow LC\_RC$

Two-Level-Regeln: Notation und Bedeutung

1.	$x{:}y \Rightarrow LC\_RC$	**nur** in diesem Kontext **darf** $x{:}y$ stehen
2.	$x{:}y \Leftarrow LC\_RC$	in diesem Kontext **darf nicht** $x{:}\neg y$ stehen
3.	$x{:}y \Leftrightarrow LC\_RC$	**nur** in diesem Kontext **darf** $x{:}y$ stehen und es **darf nicht** $x{:}\neg y$ stehen

>>>

Two-Level-Regeln: Notation und Bedeutung

1.	$x{:}y \Rightarrow LC\_RC$	**nur** in diesem Kontext **darf** *x:y* stehen
2.	$x{:}y \Leftarrow LC\_RC$	in diesem Kontext **darf nicht** $x{:}\neg y$ stehen
3.	$x{:}y \Leftrightarrow LC\_RC$	**nur** in diesem Kontext **darf** *x:y* stehen und es **darf nicht** $x{:}\neg y$ stehen

>>>

Anwendung von Two-Level-Regeln

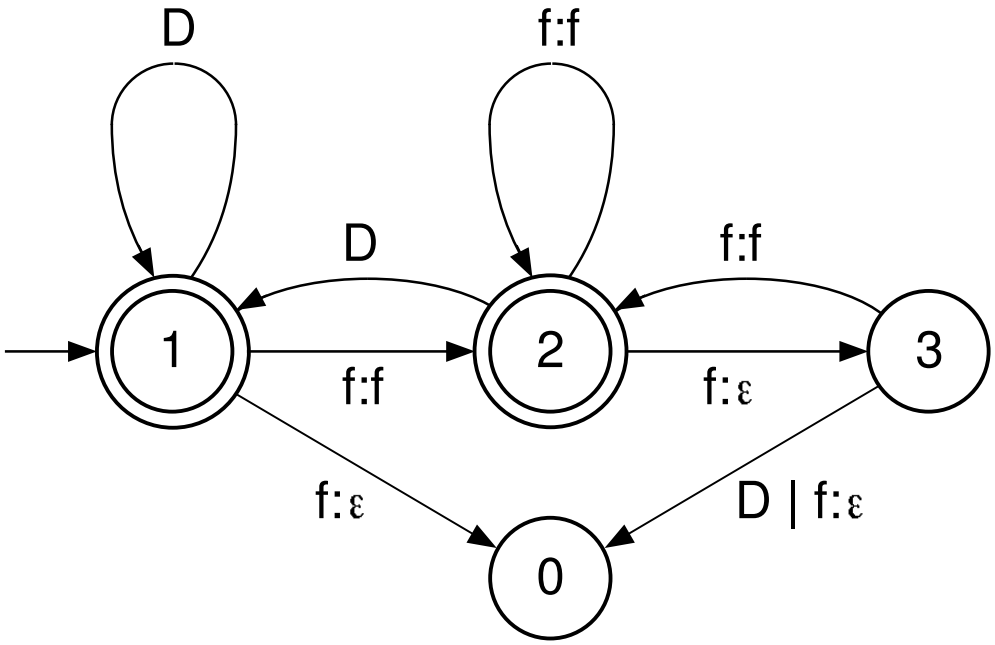
Two-Level-Regeln werden nicht direkt angewendet
(z.B. in einem Parser wie DCG-Regeln)
sondern in endliche Automaten übersetzt!

Die Eingabesymbole der Automaten sind jedoch Symbolpaare

Wie sieht der endliche Automat für die Two-Level-Regel $f{:}\varepsilon \Rightarrow f{:}f \_ f{:}f$ aus ?
(optionale Reduktion von Dreifach-F)

A_E für die Two-Level-Regel: *f:ε ⇒ f:f _ f:f*

Hinweis: Zur Sprache von A_E gehören alle Sequenzen von Symbolpaaren, die kein *f:ε* enthalten und alle mit der Folge *f:f f:ε f:f*



	D	*f:f*	*f:ε*
1	1	2	0
2	1	2	3
3	0	2	0

Einsatz des A_E als Transduktor

Umformung: Folge lexikalischer Symbole in Oberflächensymbole oder umgekehrt
d.h. nur linkes oder rechtes Symbol als Eingabesymbol betrachten,
das andere als Ausgabe

Einsatz des A_E als Transduktor

Umformung: Folge lexikalischer Symbole in Oberflächensymbole oder umgekehrt
d.h. nur linkes oder rechtes Symbol als Eingabesymbol betrachten,
das andere als Ausgabe

Merke: A_E als Umformer eingesetzt $\longrightarrow$ i.a. nicht deterministisch !

Einsatz des A_E als Transduktor

Umformung: Folge lexikalischer Symbole in Oberflächensymbole oder umgekehrt
d.h. nur linkes oder rechtes Symbol als Eingabesymbol betrachten,
das andere als Ausgabe

Merke: A_E als Umformer eingesetzt $\longrightarrow$ i.a. nicht deterministisch !

Beispiel: lexikalische Folge: `s c h i f f f a h r t`

Zustandssequenz 1: `1 1 1 1 2 2 2 1 1 1 1`

Oberflächenfolge 1: `s c h i f f f a h r t`

Einsatz des A_E als Transduktor

Umformung: Folge lexikalischer Symbole in Oberflächensymbole oder umgekehrt
d.h. nur linkes oder rechtes Symbol als Eingabesymbol betrachten,
das andere als Ausgabe

Merke: A_E als Umformer eingesetzt $\longrightarrow$ i.a. nicht deterministisch !

Beispiel:

lexikalische Folge:	s c h i f f f a h r t
Zustandssequenz 1:	1 1 1 1 2 2 2 1 1 1 1
Oberflächenfolge 1:	s c h i f f f a h r t
Zustandssequenz 2:	1 1 1 1 2 3 2 1 1 1 1
Oberflächenfolge 2:	s c h i f ε f a h r t

Übersetzen von Two-Level-Regeln in Transduktoren

1.	$x{:}y \Rightarrow LC \_ RC$	**nur** in diesem Kontext **darf** $x{:}y$ stehen	>>>
2.	$x{:}y \Leftarrow LC \_ RC$	in diesem Kontext **darf nicht** $x{:}\neg y$ stehen	>>>
3.	$x{:}y \Leftrightarrow LC \_ RC$	**nur** in diesem Kontext **darf** $x{:}y$ stehen und es **darf nicht** $x{:}\neg y$ stehen	>>>

Übersetzen von Two-Level-Regeln in Transduktoren

Gegeben: Satz von Two-Level-Regeln $R_1 \ldots R_k$
(alle sollen gleichzeitig gelten)

Gesucht: Ein Transduktor

Vorgehen:
1. jede Regel R_i in einen Transduktor T_i überführen
2. stets zwei Transduktoren mit dem Algorithmus 6.20 zu einem zusammenfassen bis nur noch einer übrig ist

Zusammenfassung

Two-Level-Regeln

- Definieren kontextabhängige Symbolersetzungen
- Lassen sich in endliche Automaten mit Eingabesymbolpaaren übersetzen
  $\longrightarrow$ Transduktoren
- Eignen sich zum Überprüfen von Folgen von Symbolpaaren und zum Umformen von Symbolfolgen
- Als Umformer eingesetzt sind Transduktoren i.a. nichtdeterministisch

Thema der nächsten Lektion

Morphologische Analyse

Automatische Transkription

Zur Übersicht der Vorlesung *Sprachverarbeitung II* >>>

Two-Level-Regel des Typs $\Rightarrow$

Beispiel: $f{:}\varepsilon \Rightarrow f{:}f \_ f{:}f$

Aussage: Nur zwischen *f:f* und *f:f* darf *f:ε* stehen
Merke: muss aber nicht!

Folge: lexikalische Ebene: s c h i f f f a h r t

Oberflächenebene: a) s c h i f f f a h r t

b) s c h i f ε f a h r t

<<<

Two-Level-Regel des Typs $\Leftrightarrow$

Beispiel: *e:ε* $\Leftrightarrow$ _ *l:l +:+ e:e*

Aussage: Nur wenn nachher die Symbolpaare *l:l +:+ e:e* folgen, muss *e:ε* stehen

Folge: lexikalische Ebene: `h a n d e l + e`

Oberflächenebene: `h a n d ε l + e`

<<<

Transduktor für eine Two-Level-Regel des Typs $\Rightarrow$

Beispiel: *x:y* $\Rightarrow$ *b:b c:c _ d:d e:e*

<<<

Transduktor für eine Two-Level-Regel des Typs $\Rightarrow$

Beispiel: *x:y* $\Rightarrow$ *b:b c:c _ d:d e:e*

Automat muss die Folge
b:b c:c x:y d:d e:e
detektieren können

<<<

Transduktor für eine Two-Level-Regel des Typs $\Rightarrow$

Beispiel: *x:y* $\Rightarrow$ *b:b c:c _ d:d e:e*

Automat muss die Folge
b:b c:c x:y d:d e:e
detektieren können

1						
2						
3						
4						
5						

<<<

Transduktor für eine Two-Level-Regel des Typs $\Rightarrow$

Beispiel: *x:y* $\Rightarrow$ *b:b c:c _ d:d e:e*

Automat muss die Folge
b:b c:c x:y d:d e:e
detektieren können

	D_1	*b:b*	*c:c*	*x:y*	*d:d*	*e:e*
1						
2						
3						
4						
5						

<<<

Transduktor für eine Two-Level-Regel des Typs $\Rightarrow$

Beispiel: *x:y* $\Rightarrow$ *b:b c:c* _ *d:d e:e*

	D_1	*b:b*	*c:c*	*x:y*	*d:d*	*e:e*
1		2				
2			3			
3				4		
4					5	
5						

<<<

Transduktor für eine Two-Level-Regel des Typs $\Rightarrow$

Beispiel: *x:y* $\Rightarrow$ *b:b c:c* _ *d:d e:e*

Eingabe *x:y* ist nur
im Zustand 3 erlaubt

	D_1	*b:b*	*c:c*	*x:y*	*d:d*	*e:e*
1		2				
2			3			
3				4		
4					5	
5						1

<<<

Transduktor für eine Two-Level-Regel des Typs $\Rightarrow$

Beispiel: *x:y* $\Rightarrow$ *b:b c:c* _ *d:d e:e*

Fehler falls nach *x:y*
nicht *d:d und e:e* folgen

	D_1	*b:b*	*c:c*	*x:y*	*d:d*	*e:e*
1		2		0		
2			3	0		
3				4		
4				0	5	
5				0		1

<<<

Transduktor für eine Two-Level-Regel des Typs $\Rightarrow$

Beispiel: *x:y* $\Rightarrow$ *b:b c:c* _ *d:d e:e*

b:b führt in den Zustand 2
(Anfang des Kontextes)

	D_1	*b:b*	*c:c*	*x:y*	*d:d*	*e:e*
1		2		0		
2			3	0		
3				4		
4	0	0	0	0	5	0
5	0	0	0	0	0	1

<<<

Transduktor für eine Two-Level-Regel des Typs $\Rightarrow$

Beispiel: *x:y* $\Rightarrow$ *b:b c:c* _ *d:d e:e*

	D_1	*b:b*	*c:c*	*x:y*	*d:d*	*e:e*
1		2		0		
2		2	3	0		
3				4		
4	0	0	0	0	5	0
5	0	0	0	0	0	1

<<<

Transduktor für eine Two-Level-Regel des Typs $\Rightarrow$

Beispiel: *x:y* $\Rightarrow$ *b:b c:c* _ *d:d e:e*

	D_1	*b:b*	*c:c*	*x:y*	*d:d*	*e:e*
1		2		0		
2		2	3	0		
3		2		4		
4	0	0	0	0	5	0
5	0	0	0	0	0	1

<<<

Transduktor für eine Two-Level-Regel des Typs $\Rightarrow$

Beispiel: *x:y* $\Rightarrow$ *b:b c:c* _ *d:d e:e*

Endzustände ?

	D_1	*b:b*	*c:c*	*x:y*	*d:d*	*e:e*
1	1	2	1	0	1	1
2	1	2	3	0	1	1
3	1	2	1	4	1	1
4	0	0	0	0	5	0
5	0	0	0	0	0	1

<<<

Transduktor für eine Two-Level-Regel des Typs $\Rightarrow$

Beispiel: *x:y* $\Rightarrow$ *b:b c:c* _ *d:d e:e*

	D_1	*b:b*	*c:c*	*x:y*	*d:d*	*e:e*
1	1	2	1	0	1	1
2	1	2	3	0	1	1
3	1	2	1	4	1	1
4	0	0	0	0	5	0
5	0	0	0	0	0	1

<<<

Transduktor für eine Two-Level-Regel des Typs $\Leftarrow$

Beispiel: *x:y* $\Leftarrow$ *b:b c:c* _ *d:d e:e*

Was muss der Automat detektieren?

<<<

Transduktor für eine Two-Level-Regel des Typs $\Leftarrow$

Beispiel: *x:y* $\Leftarrow$ *b:b c:c* _ *d:d e:e*

Automat muss die Folge
b:b c:c x:¬y d:d e:e
detektieren können

<<<

Transduktor für eine Two-Level-Regel des Typs $\Leftarrow$

Beispiel: *x:y* $\Leftarrow$ *b:b c:c* _ *d:d e:e*

Automat muss die Folge
b:b c:c x:¬y d:d e:e
detektieren können

1						
2						
3						
4						
5						

<<<

Transduktor für eine Two-Level-Regel des Typs $\Leftarrow$

Beispiel: *x:y* $\Leftarrow$ *b:b c:c* _ *d:d e:e*

Automat muss die Folge
b:b c:c x:¬y d:d e:e
detektieren können

	D_2	*b:b*	*c:c*	*x:¬y*	*d:d*	*e:e*
1						
2						
3						
4						
5						

<<<

Transduktor für eine Two-Level-Regel des Typs $\Leftarrow$

Beispiel: *x:y* $\Leftarrow$ *b:b c:c* _ *d:d e:e*

Wohin führt Eingabe *e:e*
im Zustand 5 ?

	D_2	*b:b*	*c:c*	*x:¬y*	*d:d*	*e:e*
1		2				
2			3			
3				4		
4					5	
5						

<<<

Transduktor für eine Two-Level-Regel des Typs $\Leftarrow$

Beispiel: *x:y* $\Leftarrow$ *b:b c:c* _ *d:d e:e*

Wohin führt Eingabe *b:b*?

	D_2	*b:b*	*c:c*	*x:¬y*	*d:d*	*e:e*
1		2				
2			3			
3				4		
4					5	
5						0

<<<

Transduktor für eine Two-Level-Regel des Typs $\Leftarrow$

Beispiel: *x:y* $\Leftarrow$ *b:b c:c* _ *d:d e:e*

	D_2	*b:b*	*c:c*	*x:¬y*	*d:d*	*e:e*
1		2				
2		2	3			
3		2		4		
4		2			5	
5		2				0

<<<

Transduktor für eine Two-Level-Regel des Typs $\Leftarrow$

Beispiel: *x:y* $\Leftarrow$ *b:b c:c* _ *d:d e:e*

	D_2	*b:b*	*c:c*	*x:¬y*	*d:d*	*e:e*
1	1	2	1	1	1	1
2	1	2	3	1	1	1
3	1	2	1	4	1	1
4	1	2	1	1	5	1
5	1	2	1	1	1	0

<<<

Transduktor für eine Two-Level-Regel des Typs $\Leftrightarrow$

Regelbeispiel: *x:y* $\Leftrightarrow$ *LC* _ *RC*
(Variante $\Rightarrow$ und $\Leftarrow$ gelten gleichzeitig)

Sprache von $T_{\Rightarrow}$:

Sprache von $T_{\Leftarrow}$:

<<<

Transduktor für eine Two-Level-Regel des Typs $\Leftrightarrow$

Regelbeispiel: *x:y* $\Leftrightarrow$ *LC* _ *RC*

(Variante $\Rightarrow$ und $\Leftarrow$ gelten gleichzeitig)

Sprache von $T_{\Rightarrow}$: $L(T_{\Rightarrow}) = D_0^* \cup \{w \mid w$ enthält *LC x:y RC*$\}$

D_0: Menge aller Symbolpaare ausser *x:y*

Sprache von $T_{\Leftarrow}$:

<<<

Transduktor für eine Two-Level-Regel des Typs $\Leftrightarrow$

Regelbeispiel: *x:y* $\Leftrightarrow$ *LC* _ *RC*
(Variante $\Rightarrow$ und $\Leftarrow$ gelten gleichzeitig)

Sprache von $T_{\Rightarrow}$: $L(T_{\Rightarrow}) = D_0^* \cup \{w \mid w$ enthält *LC x:y RC*$\}$

D_0: Menge aller Symbolpaare ausser *x:y*

Sprache von $T_{\Leftarrow}$: $L(T_{\Leftarrow}) = D_1^* \setminus \{w \mid w$ enthält *LC x:¬y RC*$\}$

D_1: Menge aller Symbolpaare (inkl. *x:y*), also ist $D_1 = D_0 \cup \{$*x:y*$\}$

Sprache von $T_{\Leftrightarrow}$:

<<<

Transduktor für eine Two-Level-Regel des Typs $\Leftrightarrow$

Regelbeispiel: *x:y* $\Leftrightarrow$ *LC* _ *RC*
(Variante $\Rightarrow$ und $\Leftarrow$ gelten gleichzeitig)

Sprache von $T_{\Rightarrow}$: $L(T_{\Rightarrow}) = D_0^* \cup \{w \mid w$ enthält *LC x:y RC*$\}$
D_0: Menge aller Symbolpaare ausser *x:y*

Sprache von $T_{\Leftarrow}$: $L(T_{\Leftarrow}) = D_1^* \setminus \{w \mid w$ enthält *LC x:¬y RC*$\}$
D_1: Menge aller Symbolpaare (inkl. *x:y*), also ist $D_1 = D_0 \cup \{$*x:y*$\}$

Sprache von $T_{\Leftrightarrow}$: $L(T_{\Leftrightarrow}) = L(T_{\Rightarrow}) \cap L(T_{\Leftarrow})$
$= \{w \mid w \in L(T_{\Rightarrow})$ **und** $w \in L(T_{\Leftarrow}\})$

<<<

Transduktor für eine Two-Level-Regel des Typs $\Leftrightarrow$

Regelbeispiel: *x:y* $\Leftrightarrow$ *LC* _ *RC*
(Variante $\Rightarrow$ und $\Leftarrow$ gelten gleichzeitig)

Sprache von $T_{\Rightarrow}$: $L(T_{\Rightarrow}) = D_0^* \cup \{w \mid w$ enthält *LC x:y RC*$\}$
D_0: Menge aller Symbolpaare ausser *x:y*

Sprache von $T_{\Leftarrow}$: $L(T_{\Leftarrow}) = D_1^* \setminus \{w \mid w$ enthält *LC x:¬y RC*$\}$
D_1: Menge aller Symbolpaare (inkl. *x:y*), also ist $D_1 = D_0 \cup \{$*x:y*$\}$

Sprache von $T_{\Leftrightarrow}$: $L(T_{\Leftrightarrow}) = L(T_{\Rightarrow}) \cap L(T_{\Leftarrow})$
$= \{w \mid w \in L(T_{\Rightarrow})$ **und** $w \in L(T_{\Leftarrow}\})$

Folge: $T_{\Leftrightarrow}$ konstruieren mit Algorithmus 6.20 aus $T_{\Rightarrow}$ und $T_{\Leftarrow}$

<<<