

Computergestützte Verifikation

2.5.03

Model Checking für finite

state systems

explizit: Kapitel 3

symbolisch: Kapitel 4

3.1: Tiefensuche

3.2: LTL-Model Checking

3.3: CTL-Model Checking

3.4: Fairness

3.5: Reduktion durch
Symmetrie

3.6: Partial Order Reduction

3.7: Tools

4.1: BDD-basiertes

CTL-Model Checking

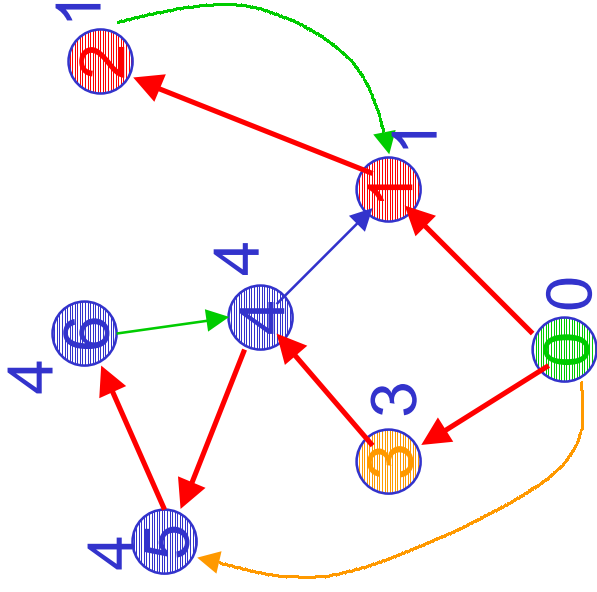
4.2: SAT-basiertes

Model Checking

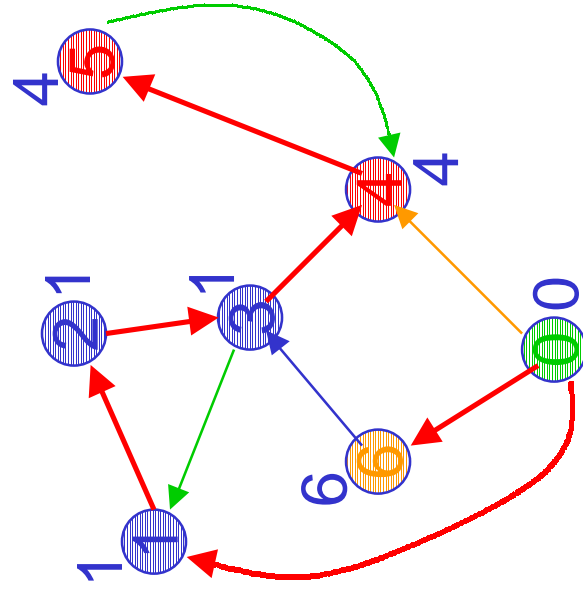
4.3: Tools

Kriterium für Startknoten

von SZK



$v.\text{lowlink} = \text{MIN}(v'.\text{dfs} \mid v' \text{ von } v \text{ erreichbar über beliebig viele Baumkanten, gefolgt von max. einer anderen Kante } [v, v'] \text{ mit } v \sim v')$



Satz: v ist genau dann Startknoten einer SZK wenn $v.\text{lowlink} = v.\text{dfs}$

LTL Model Checking

$$L_{TS}^\omega \subseteq L_\phi^\omega$$

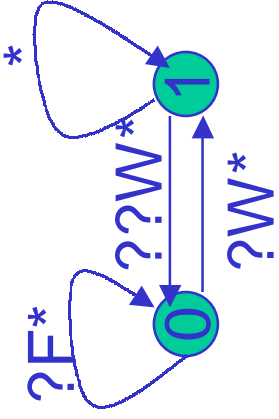
$$\Leftrightarrow L_{TS}^\omega \cap \overline{L_\phi^\omega} = \emptyset$$

$$\Leftrightarrow L_{TS}^\omega \cap L_{\neg\phi}^\omega = \emptyset$$

- Agenda:
1. Automaten, die L_{TS}^ω und $L_{\neg\phi}^\omega$ akzeptieren
 2. Konstruktion eines Schnittautomaten
 3. Entscheidung, ob Automat die leere Sprache akzeptiert

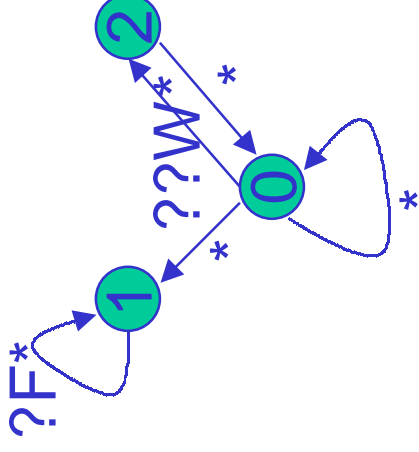
Problem: Wir reden über **unendliche** Sequenzen!

Automaten für komplexe Formeln (Bsp)



$G(a_2 \Rightarrow F a_3)$

$Z_0 = \{0\}$
 $F = \{\{0\}\}$



$(G F a_2) \Rightarrow (G F a_3)$

$Z_0 = \{0\}$
 $F = \{\{1,2\}\}$

Zwischenfazit

Haben:

- Büchi-Automat für L_{TS}
- Büchi-Automat für $L_{\neg\phi}$

Ziel: $L_{TS} \cap L_{\neg\phi} = \emptyset$?

nächster Schritt: geg: B_1, B_2
ges: $B^* : L_{B^*} = L_{B_1} \cap L_{B_2}$

Produktautomat

Idee: beide Automaten arbeiten parallel, Akzeptierungsmengen werden nebeneinandergestellt

$$B^1 = [X, Z^1, Z_0^1, \delta^1, F^1] \quad B^2 = [X, Z^2, Z_0^2, \delta^2, F^2]$$

$$B^* = [X, Z^*, Z_0^*, \delta^*, F^*]$$

$$Z^* = Z^1 \times Z^2$$

$$Z_0^* = Z_0^1 \times Z_0^2$$

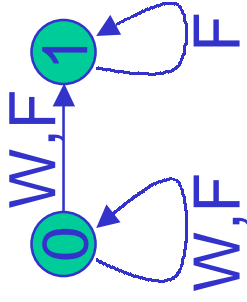
$$\delta^*([z, z'], x) = \delta^1(z, x) \times \delta^2(z', x)$$

$$F^* = \{ F \times Z^2 \mid F \in F^1 \} \cup \{ Z^1 \times F \mid F \in F^2 \}$$

Satz: $L_{B^*} = L_{B^1} \cap L_{B^2}$

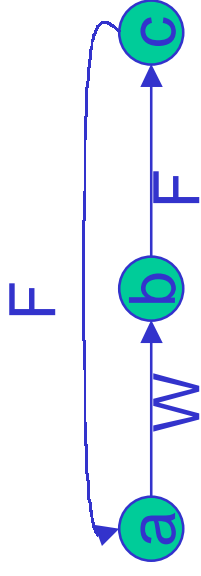
Beispiel

B1:



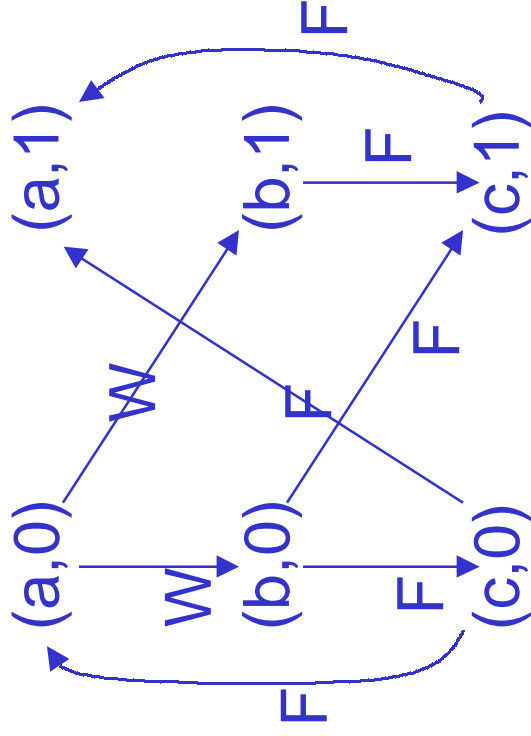
$Z_0 = \{0\}$
 $F = \{\{1\}\}$

B2:



$Z_0 = \{a\}$
 $F = \{\{a,b,c\}\}$

B* :



$Z_0 = \{(a,0)\}$ $F = \{Z, \{(a,1),(b,1),(c,1)\}\}$

Emptiness

geg.: Büchi-Automat B Frage $L_B = \emptyset$?

Lösung: $L_B \neq \emptyset$ gdw. es gibt eine nichttriv. SZK in B (von einem Initialzustand erreichbar), die aus jeder Akzeptierungsmenge Elemente enthält.

“ \Rightarrow ”: Sei π Sequenz, die von B akzeptiert wird.
 Z^* sei Menge der Zustände, die im akzeptierenden Lauf unendlich oft durchlaufen werden. Z^* ist stark zusammenhängend, und enthält Elemente aus jeder Akzeptierungsmenge

“ \Leftarrow ”: Konstruiere zur SZK eine Sequenz, die vom Initialzustand zur SZK gelangt, und dort zyklisch alle Zst. durchläuft. Diese Sequenz ist offenbar in L_B

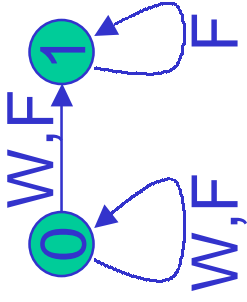
LTL Model Checking

Geg.: TS (implizit), ϕ .

1. Konstruiere $B_{\neg\phi}$
2. Konstruiere Produktautomat B^* aus TS und $B_{\neg\phi}$
3. Suche in B^* nach SZK, die aus jeder Akzeptierungsmenge ein Element enthalten
1 Schritt!
4. gefunden \rightarrow "nein", bilde Gegenbeispiel aus gefundener SZK
5. nicht gefunden \rightarrow "ja"

$O(2^{|\phi|} |TS|)$

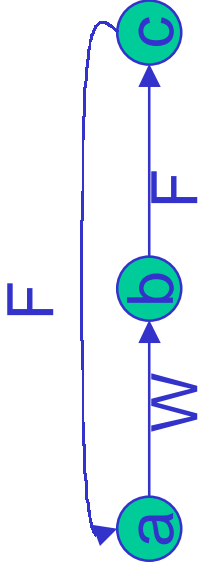
$G \models a$ (negiert: $F \models G \neg a$)



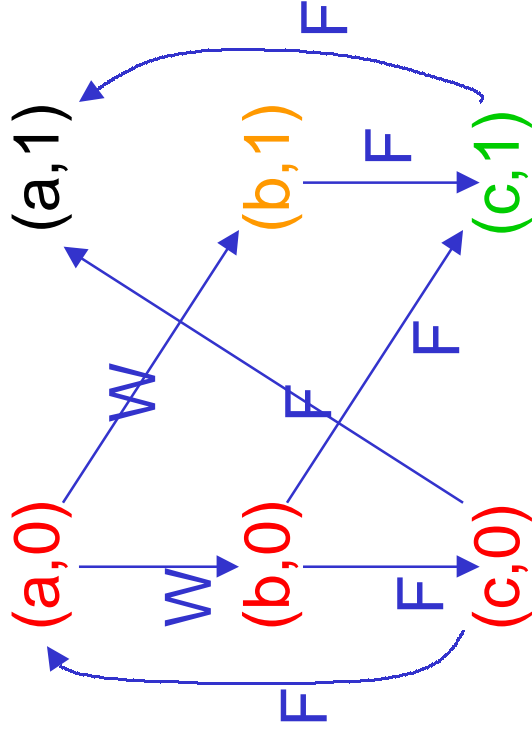
$Z_0 = \{0\}$
 $F = \{ \{1\} \}$

Beispiel

TS:



$Z_0 = \{a\}$
 $F = \{ \{a,b,c\} \}$



B^* :

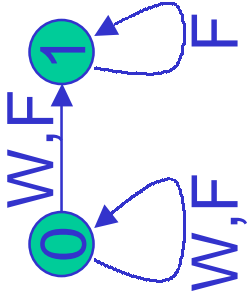
$Z_0 = \{(a,0)\}$ $F = \{ Z, \{(a,1), (b,1), (c,1)\} \}$

nur triv. SZK akzeptieren

also: Formel wahr

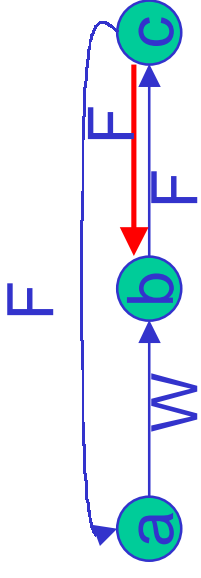
G F a (negiert: F G \neg a)

Beispiel

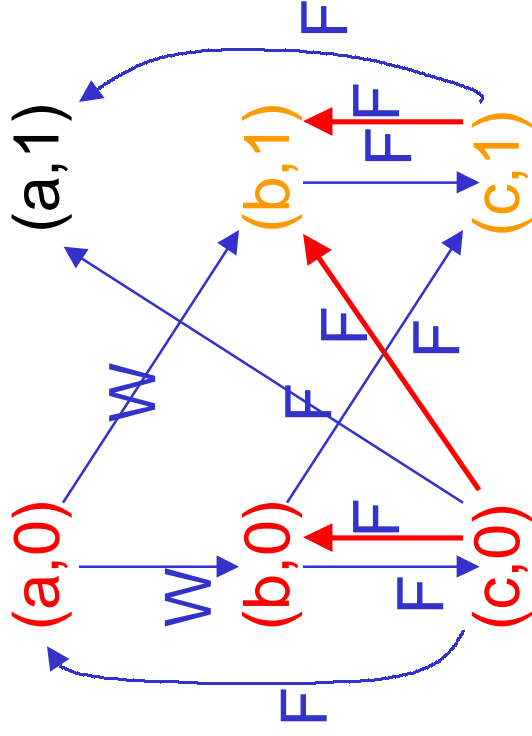


Z0 = {0}
F = { {1} }

TS:



Z0 = {a}
F = { {a,b,c} }



B* :

Z0 = {(a,0)} F = { Z, {(a,1),(b,1),(c,1)} }

nichttriv. SZK,

→ Formel falsch,

→ Gbsp: a (b c)*

On-The-Fly Verifikation

LTL-Formel wahr → kompletter Produktautomat
wird konstruiert

LTL-Formel falsch → Konstruktion des Produktautomaten
kann vorzeitig abgebrochen werden

→ weniger Speicherverbrauch

3.3 CTL Model Checking

Spezifisch für LTL: alle Formeln betreffen Pfade

Spezifisch für CTL: alle (Teil-)Formeln betreffen Zustände

Idee:

- Jeder Zustand hat Liste mit Wahrheitswerten aller Teilformeln ($\in \{W, F, ?\}$) $\rightarrow L(s, \phi)$
- Für die Wertberechnung werden Teilformeln jeweils als atomar angesehen
- Werte von Teilformeln werden bei Bedarf berechnet

Rahmenprozedur

```
CTL(s,φ)
IF L(s,φ) ≠ ? THEN RETURN END
CASE φ
∈AP:   berechne L(s,φ) END
ψ∧χ:   CTL(s,ψ)
        IF L(s,ψ) THEN
            CTL(s,χ); L(s,φ) = L(s,χ);
        ELSE
            L(s,φ) = F
        END
¬ψ, ψ∨χ: analog
AX ψ:   FOR ALL s': [s,s'] ∈ E DO
            CTL(s',ψ);
            IF L(s',ψ) = F THEN L(s,φ) = F; RETURN; END
        END
L(s,φ) = W
EX ψ:   analog
/* Fortsetzung folgt */
```

Rahmenprozedur (Fortsetzung)

(... CASE ϕ)

$E(\psi \cup \chi)$: CheckEU(s, ψ, χ);

$A(\psi \cup \chi)$: CheckAU(s, ψ, χ);

EF ψ ,

AF ψ ,

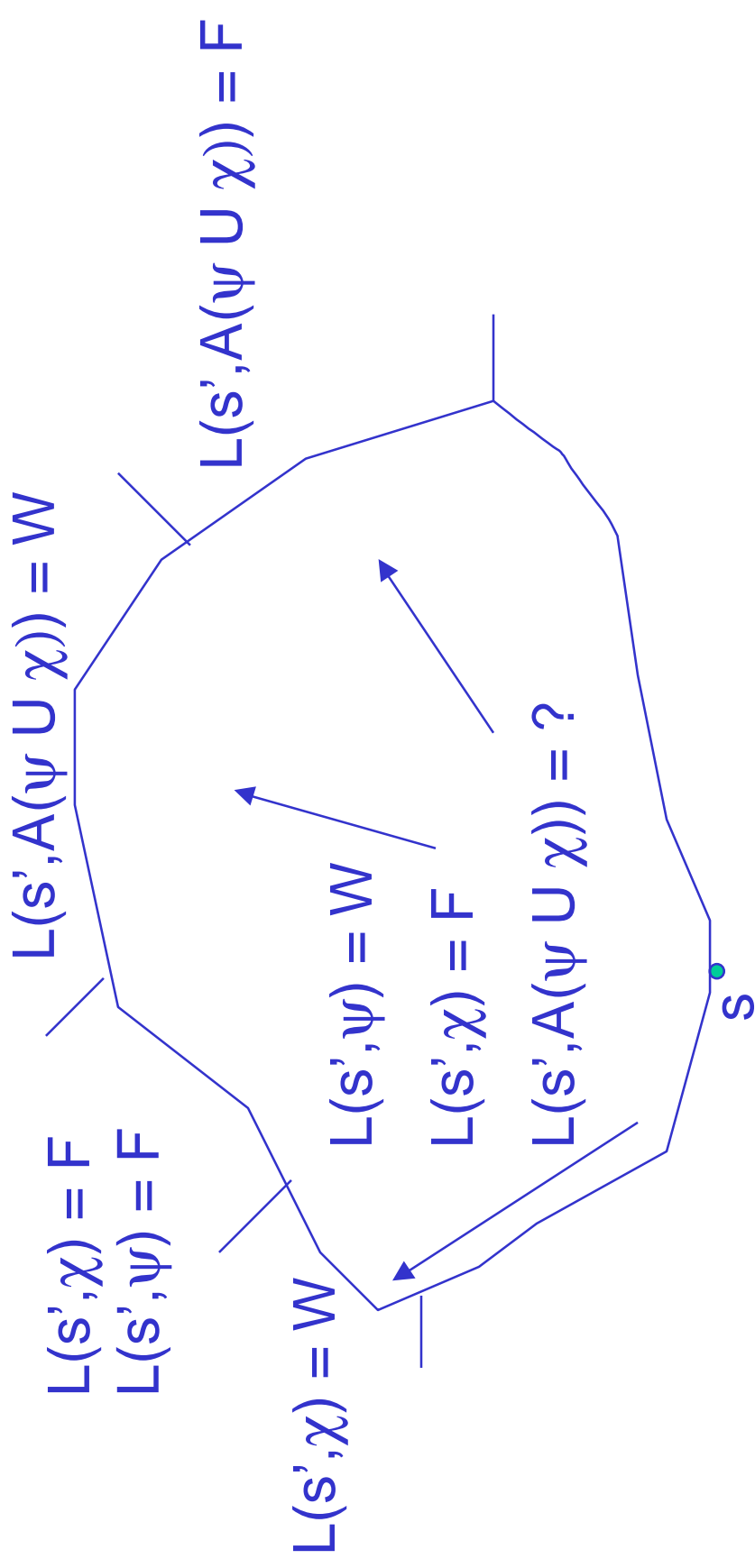
EG ψ ,

AG ψ : /* über Tautologien */

Also bleiben: CheckAU, CheckEU

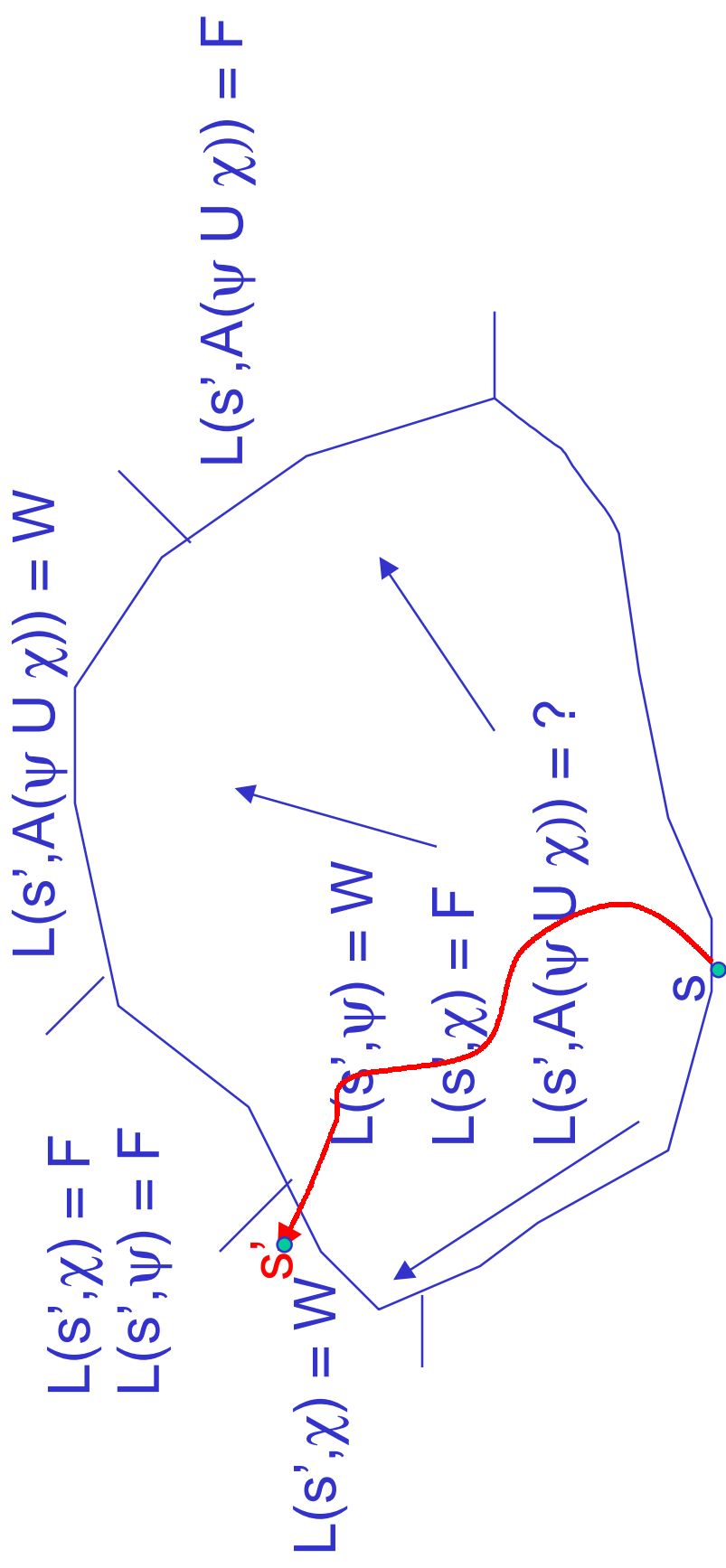
CheckAU

CheckAU(s, ψ, χ): Suche Gegenbeispiel



CheckAU

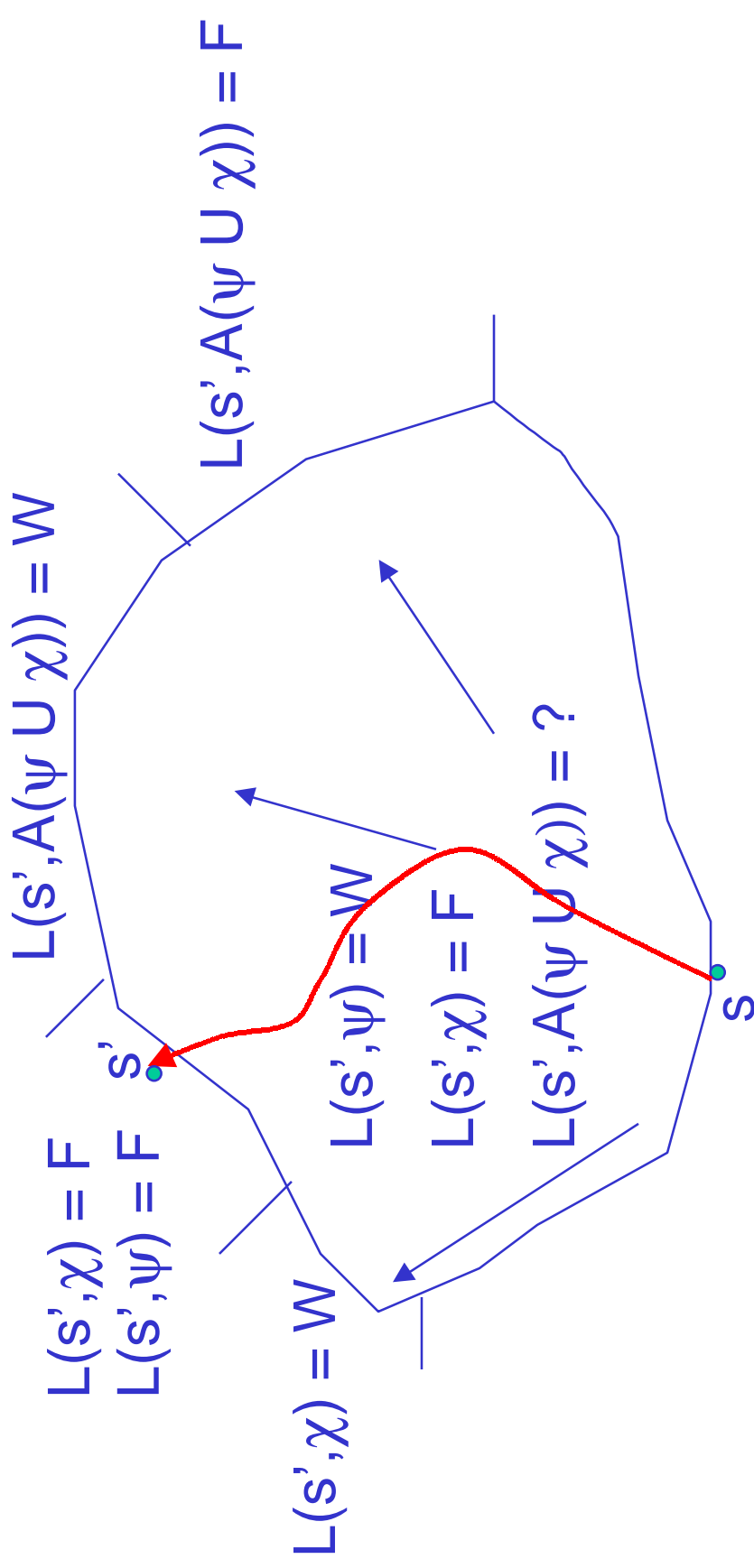
CheckAU(s, ψ, χ): Suche Gegenbeispiel



Keine Fortsetzung des Pfades kann Gegenbeispiel sein
 → Backtracking von s'

CheckAU

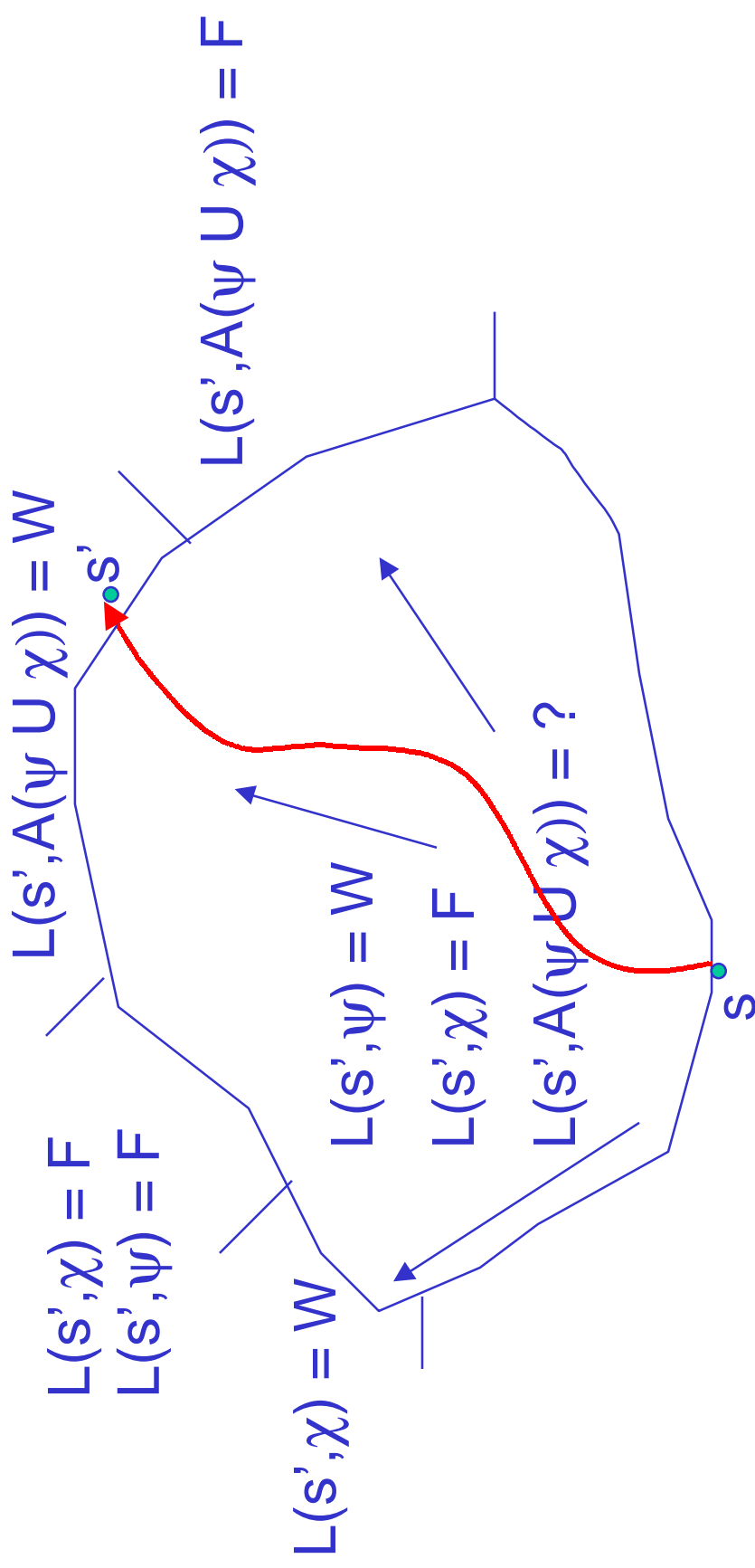
CheckAU(s, ψ, χ): Suche Gegenbeispiel



Gegenbeispiel gefunden $\rightarrow L(s, A(\psi \cup \chi)) := F$, RETURN

CheckAU

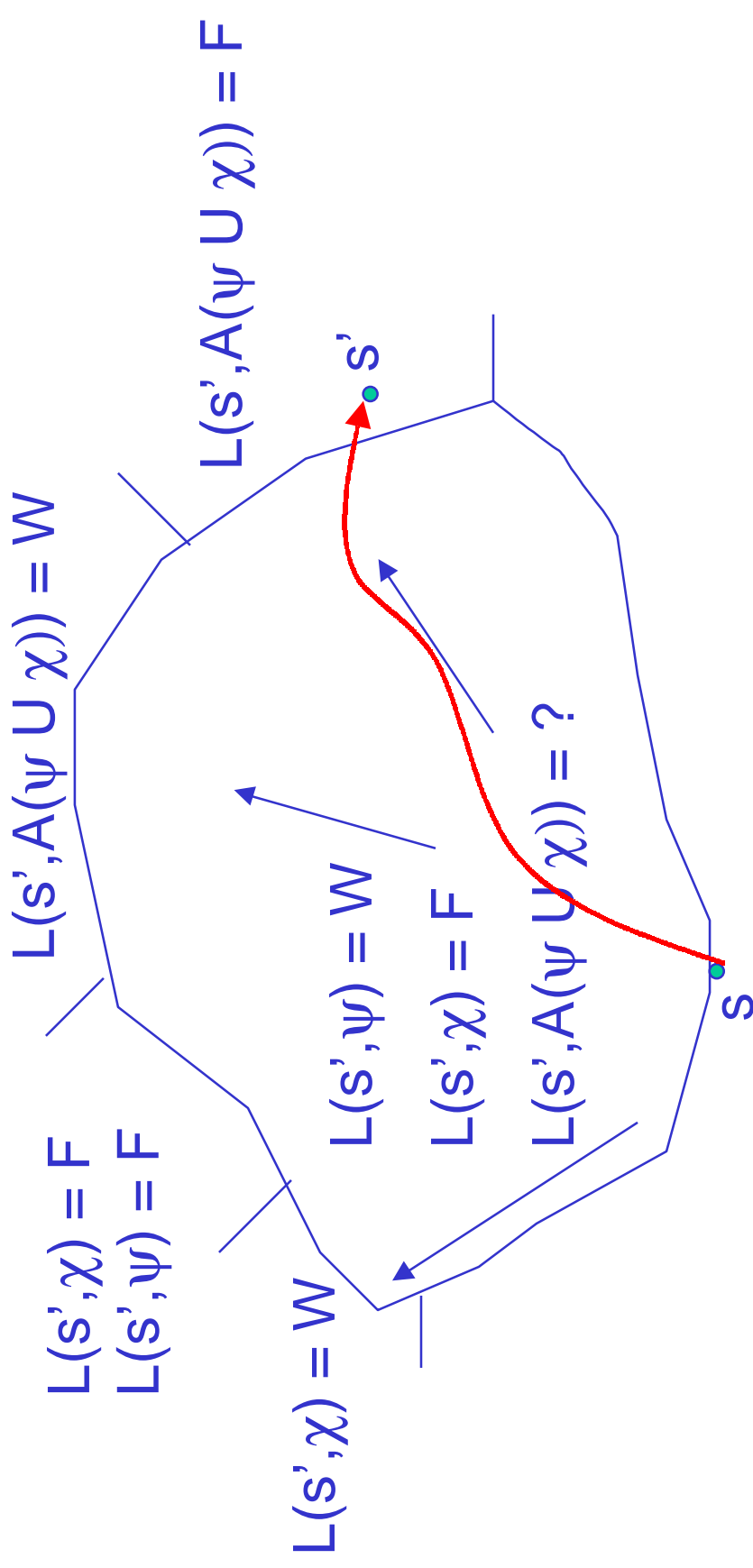
CheckAU(s, ψ, χ): Suche Gegenbeispiel



Da jede Fortsetzung bei $s' \psi \cup \chi$ erfüllt, kann keine Fortsetzung
Gegenbeispiel liefern \rightarrow Backtracking

CheckAU

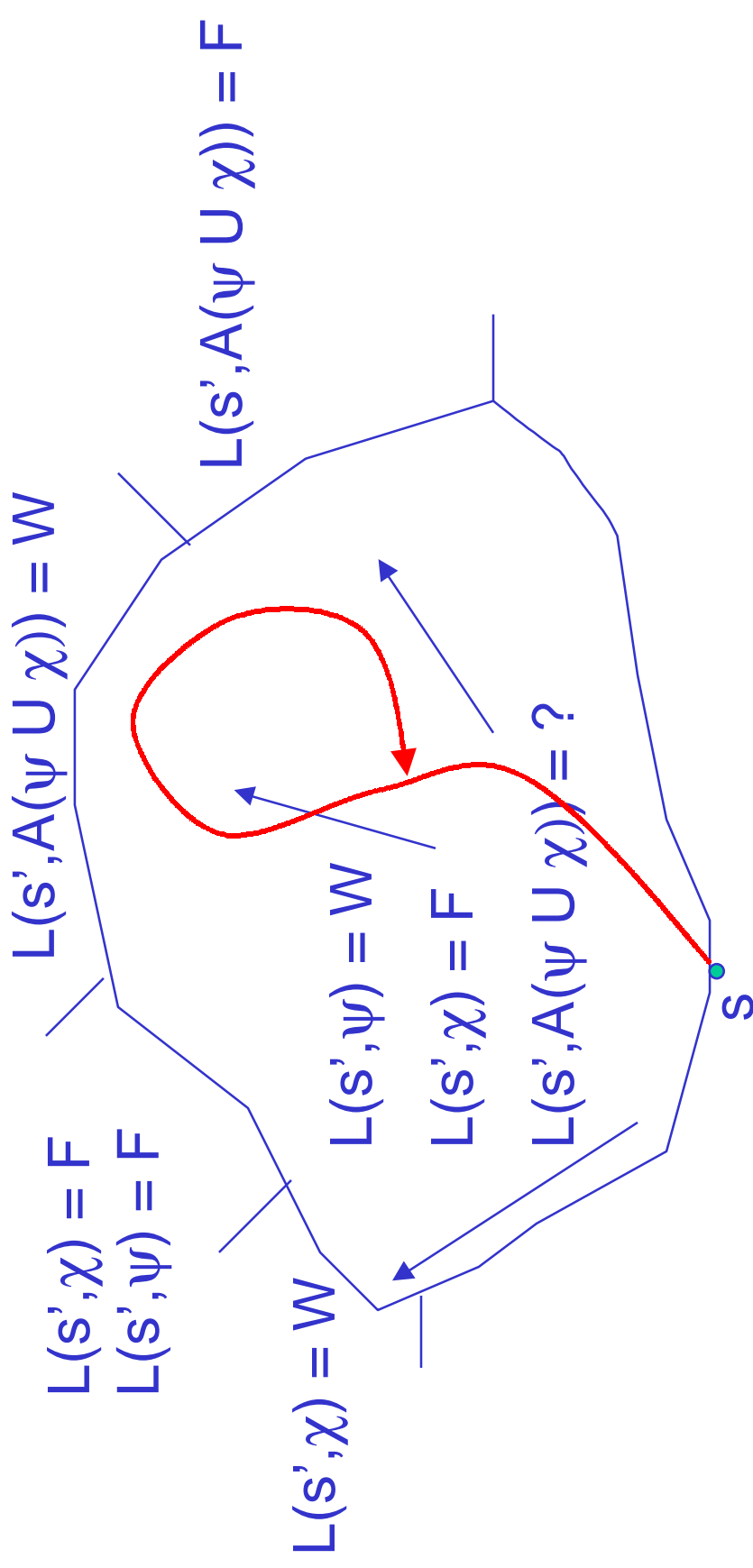
CheckAU(s, ψ, χ): Suche Gegenbeispiel



Gegenbeispiel gefunden ($= s \dots s' +$ Gegenbeispiel bei s')
 $\rightarrow L(s, A(\psi \cup \chi)) = F$, RETURN

CheckAU

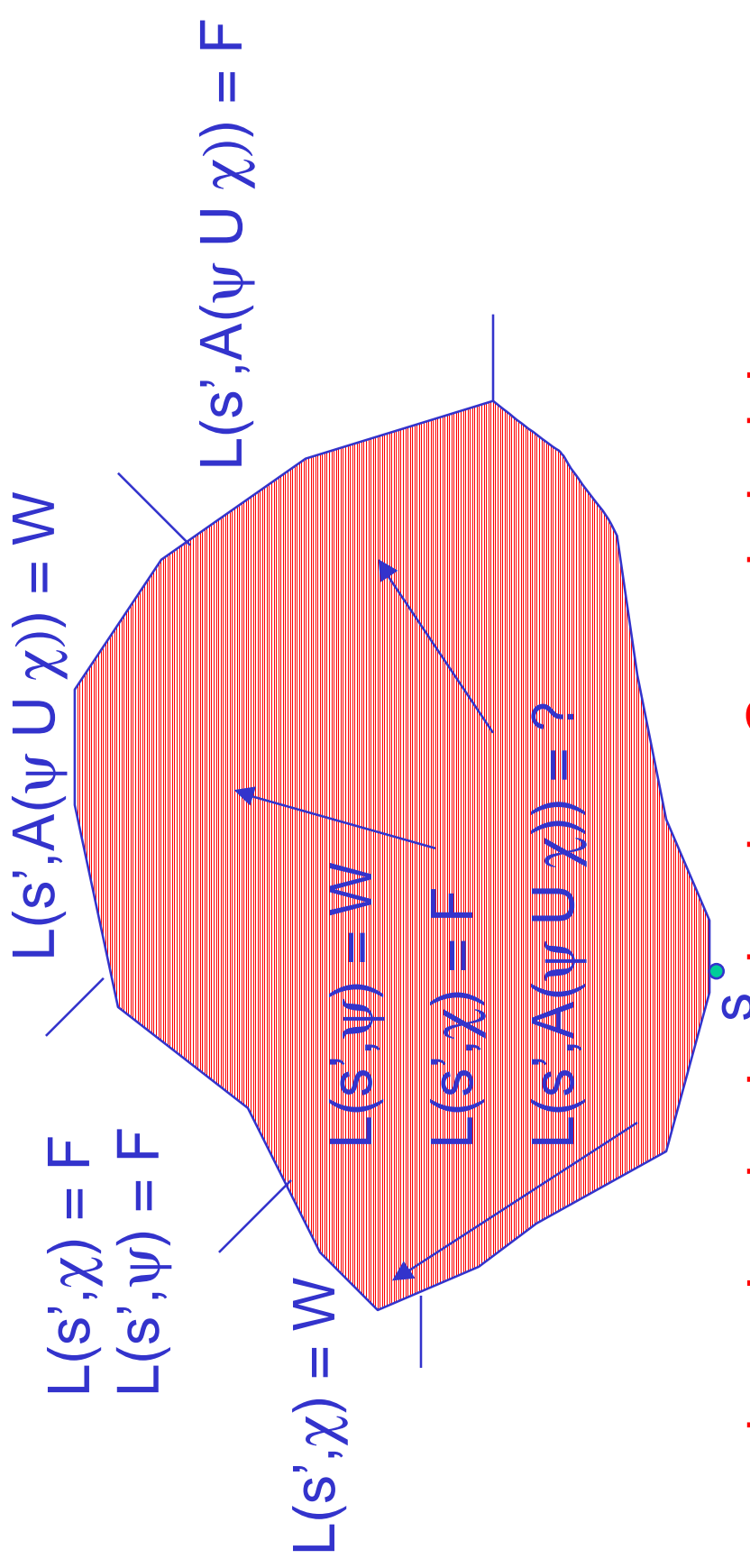
CheckAU(s, ψ, χ): Suche Gegenbeispiel



Gegenbeispiel $\rightarrow L(s, A(\psi \cup \chi)) = F$, RETURN

CheckAU

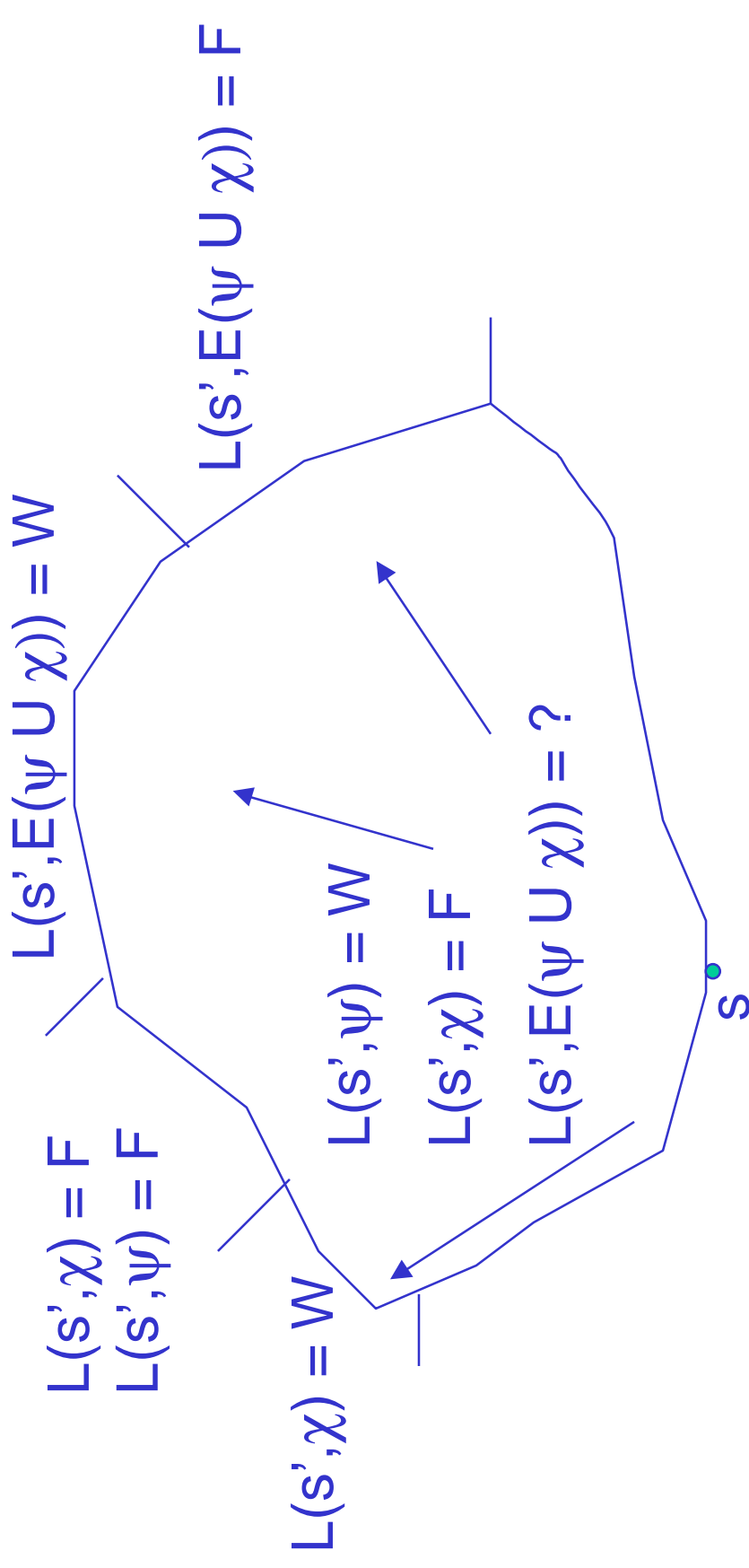
CheckAU(s, ψ, χ): Suche Gegenbeispiel



Suchraum komplett durchsucht, ohne Gegenbeispiel
 $\rightarrow L(s, A(\psi \cup \chi)) = W$

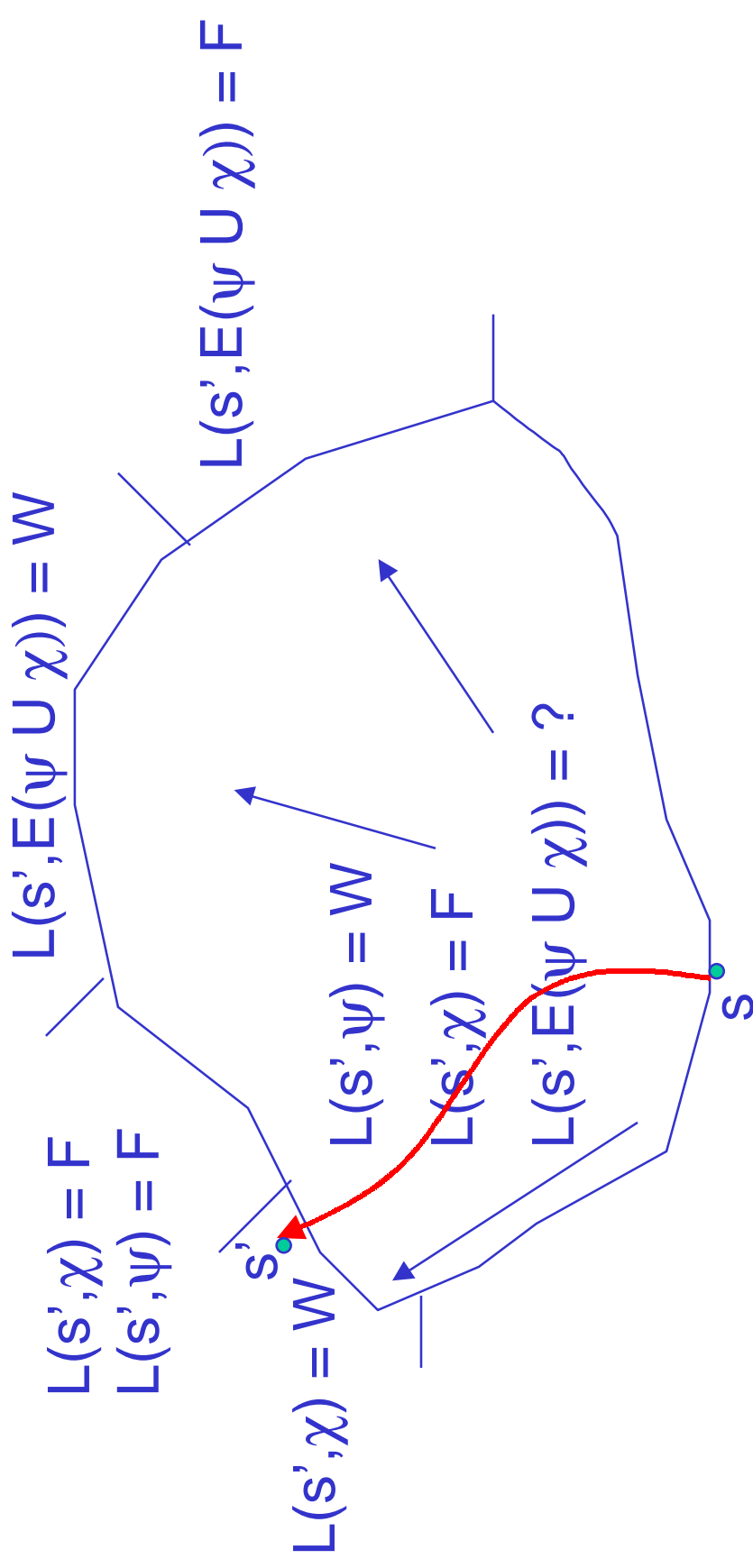
CheckEU

CheckEU(s, ψ, χ): Suche Zeugenpfad



CheckEU

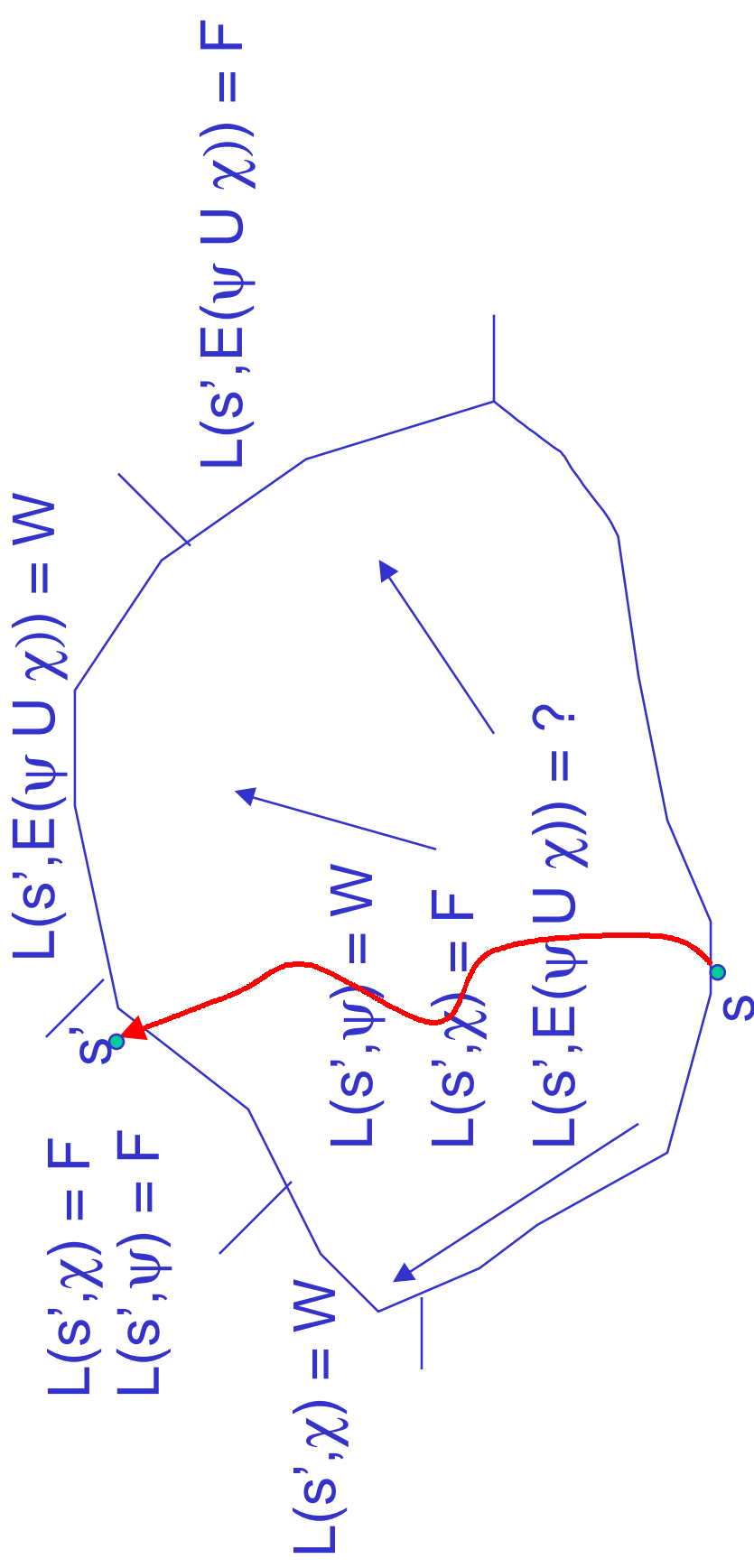
CheckEU(s, ψ, χ): Suche Zeugenpfad



Zeuge gefunden $\rightarrow L(s', E(\psi \cup \chi)) = W$, RETURN

CheckEU

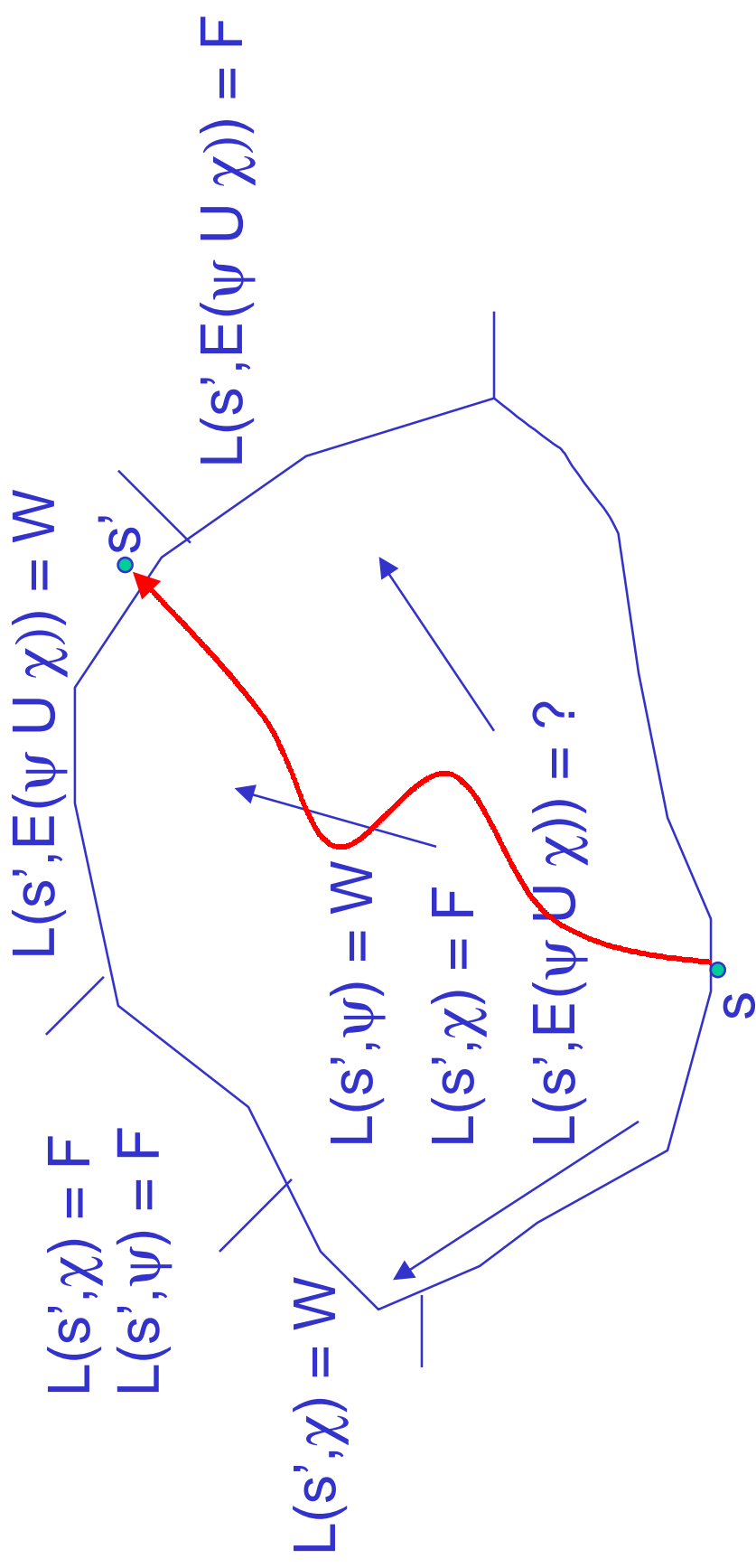
CheckEU(s, ψ, χ): Suche Zeugenpfad



keine Fortsetzung kann Zeuge sein → Backtracking

CheckEU

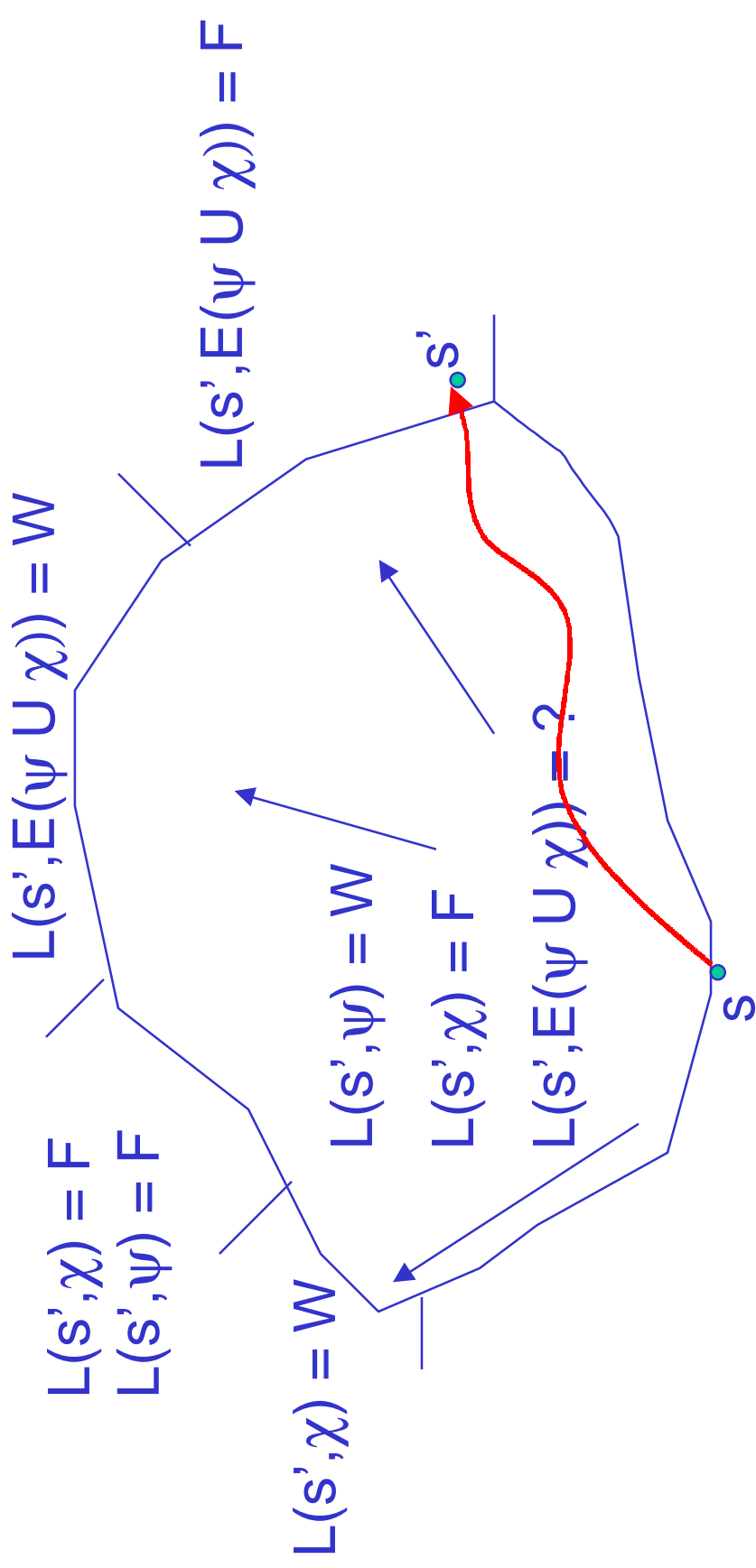
CheckEU(s, ψ, χ): Suche Zeugenpfad



Zeuge (= $s \dots s'$ + Zeuge bei s') $\rightarrow L(s', E(\psi \cup \chi)) = W$, RETURN

CheckEU

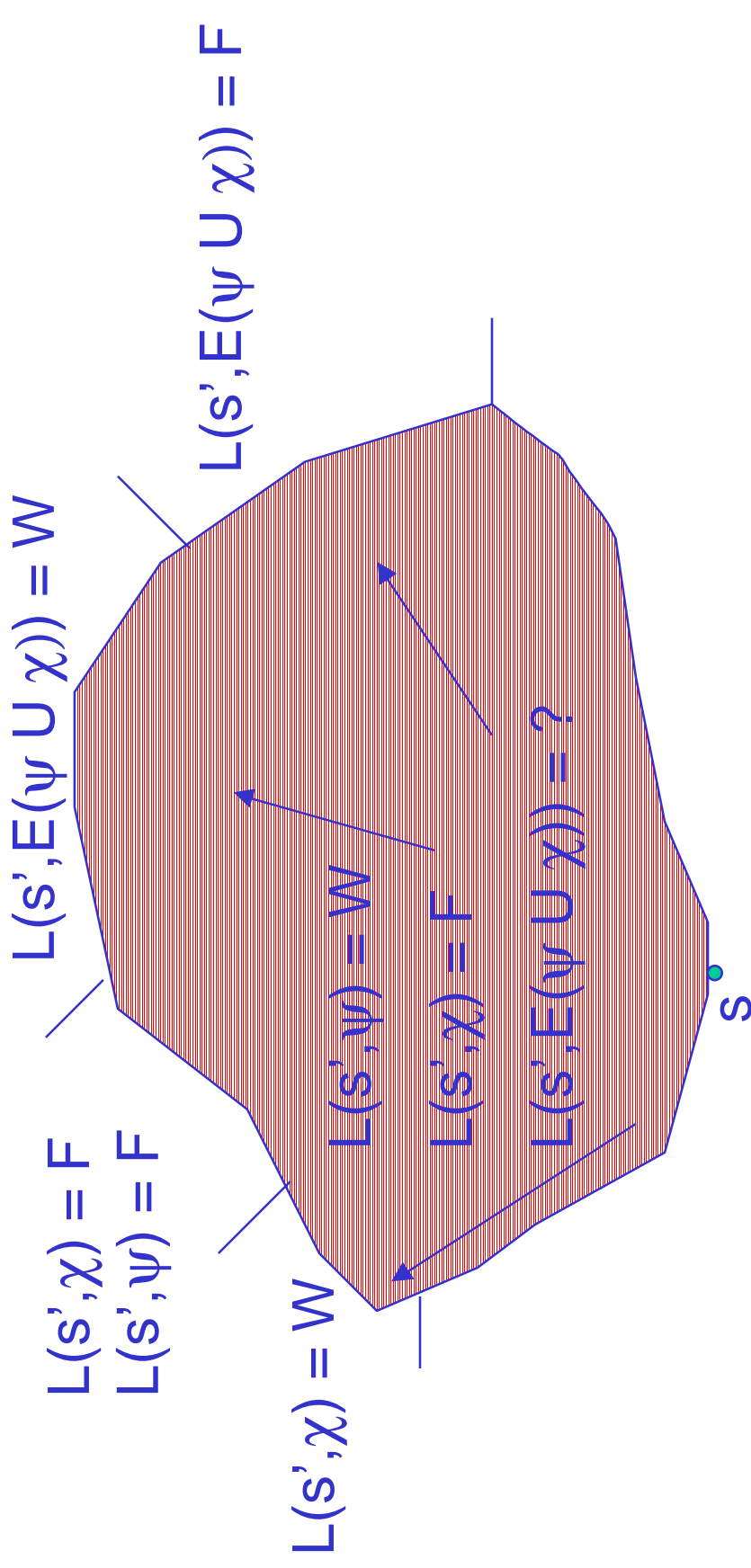
CheckEU(s, ψ, χ): Suche Zeugenpfad



Keine Fortsetzung kann Zeuge sei \rightarrow Backtracking

CheckEU

CheckEU(s, ψ, χ): Suche Zeugenpfad



Suche komplett ohne Zeuge $\rightarrow L(s', E(\psi \cup \chi)) = F$

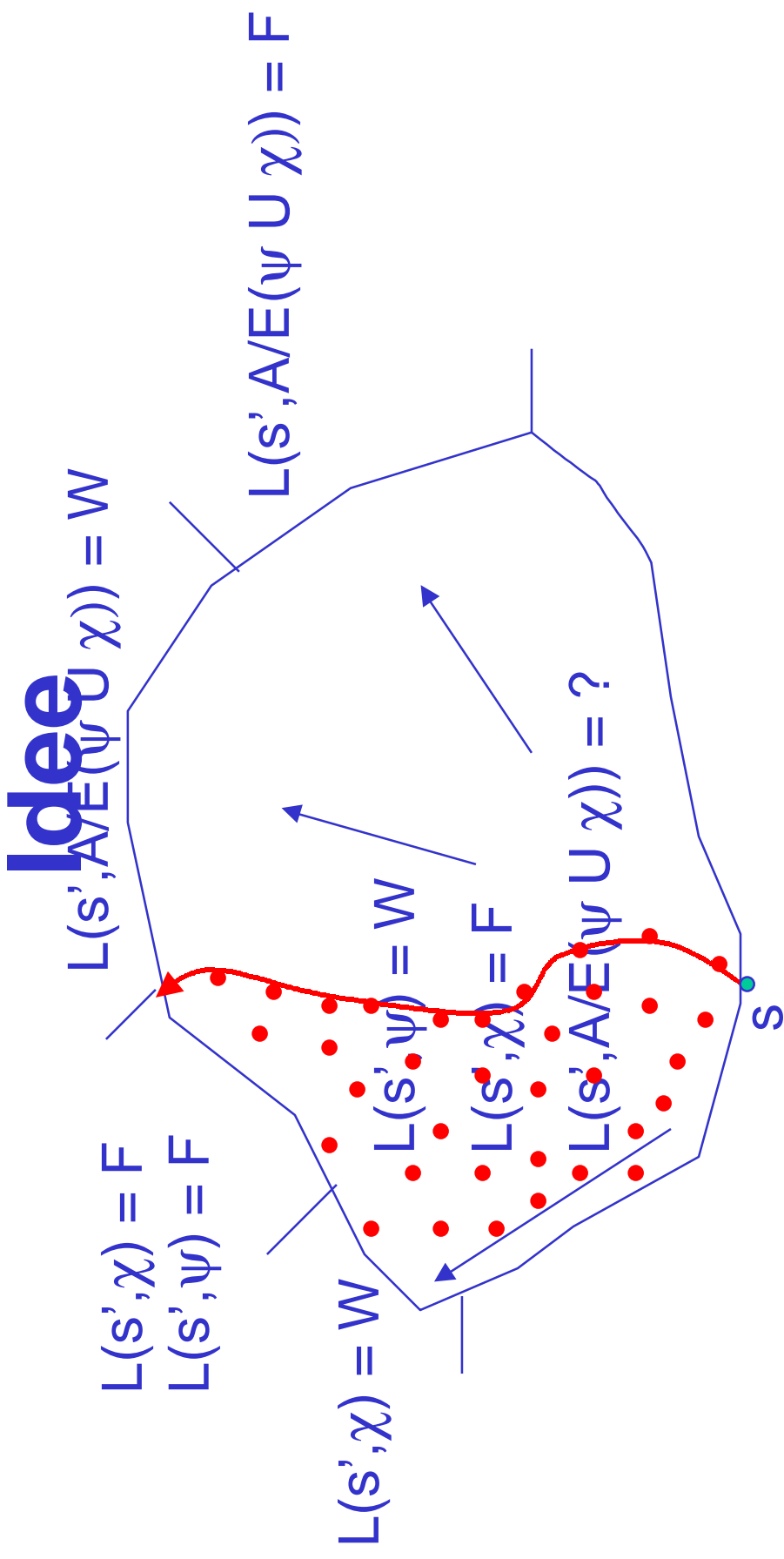
Zwischenfazit CheckAU und CheckEU

$L(s, A/E(\psi \cup \chi))$ kann durch einfache Tiefensuche ermittelt werden

Das würde Laufzeit von $O(|\phi| |TS| |TS|) = O(|\phi| |TS|^2)$ liefern: Für jede Teilformel und jeden Zustand eine Tiefensuche

Ziel: $O(|\phi| |TS|)$

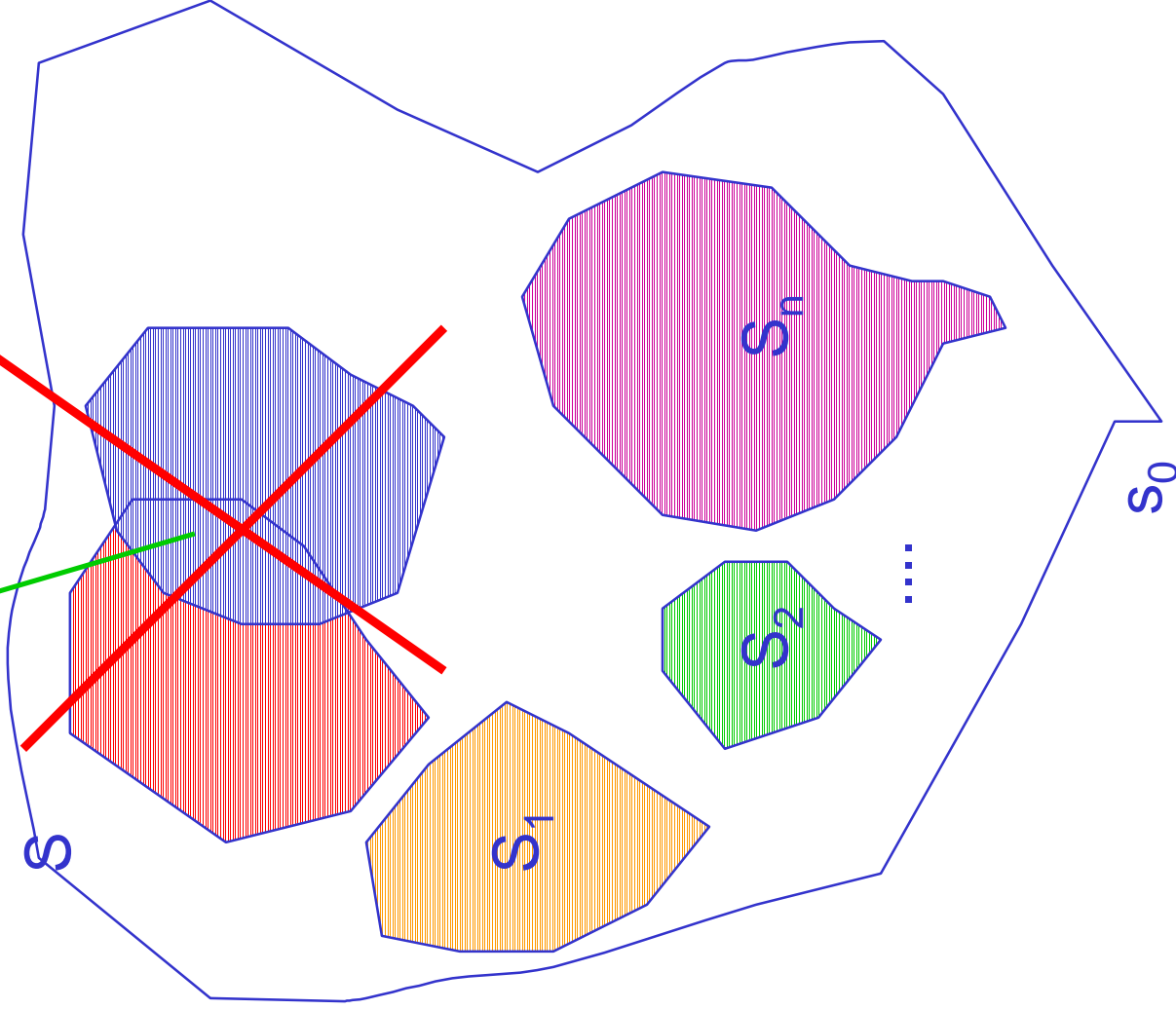
Lineare Gesamtlaufzeit -



Bestimme durch eine einzige Tiefensuche nicht nur $L(s, A/E(\psi \cup \chi))$, sondern

... auch $L(s', A/E(\psi \cup \chi))$ für *alle* während der Suche betretenen Zustände!

$L(s, A/E(\psi \cup \chi)) \neq ?$ Was hilft das?

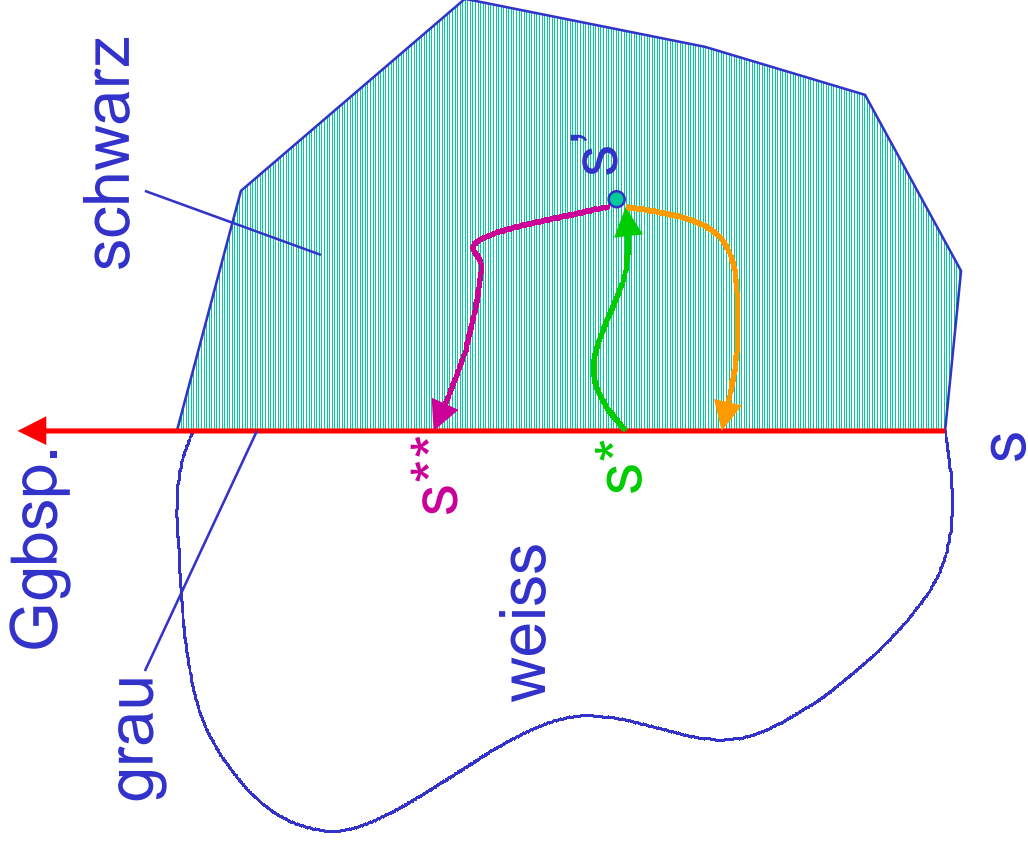


$$\begin{aligned}
 |\phi| & \left(\begin{aligned} & O(|S_1|) \\ & + O(|S_2|) \\ & + \dots \\ & + O(|S_n|) \end{aligned} \right) \\
 & = O(|\phi|(|S_1| + |S_2| + \dots + |S_n|)) \\
 & = O(|\phi| |TS|)
 \end{aligned}$$

Verbesserung von CheckAU

1. Fall $s \models A (\psi \cup \chi)$
 \Rightarrow alle s' im Suchraum: $s' \models A (\psi \cup \chi)$
(sonst: $s \dots s'$ + Gegenbeispiel bei s' wäre Gegenbsp. bei s !)
2. Fall $s \not\models A (\psi \cup \chi) \rightarrow$ Stack zum Abbruchzeitpunkt
bildet Gegenbeispiel
 - 2a) s' auf dem Tiefensuchstack (“grau”)
 \rightarrow Stack ab s' ist immer noch Gegenbeispiel, also
 $s' \not\models A (\psi \cup \chi)$
 - 2b) s' nicht mehr auf dem Tiefensuchstack (“schwarz”)
 $\rightarrow s' \models A (\psi \cup \chi)$ (b.w.)

Verbesserung von CheckAU



1. Ggbsp. ab s' ohne graue Knoten geht nicht, hätte ja bei dfs(s') gefunden werden müssen

2. wissen: ex. Weg von $s \rightarrow s'$

3. Ggbsp. ab s' muß also graue Knoten enthalten

3a. vor $s^* \rightarrow$ Kreis \rightarrow Ggbsp, wäre früher gefunden worden

3b. nach $s^* \rightarrow$ Knoten ab s^{**} wären schon bei Suche von s' aus bearbeitet worden (\rightarrow Wid!)

also: es kann kein Ggbsp. ab s' geben $\rightarrow s' \models A (\psi \cup \chi)$

CheckAU - Pseudocode

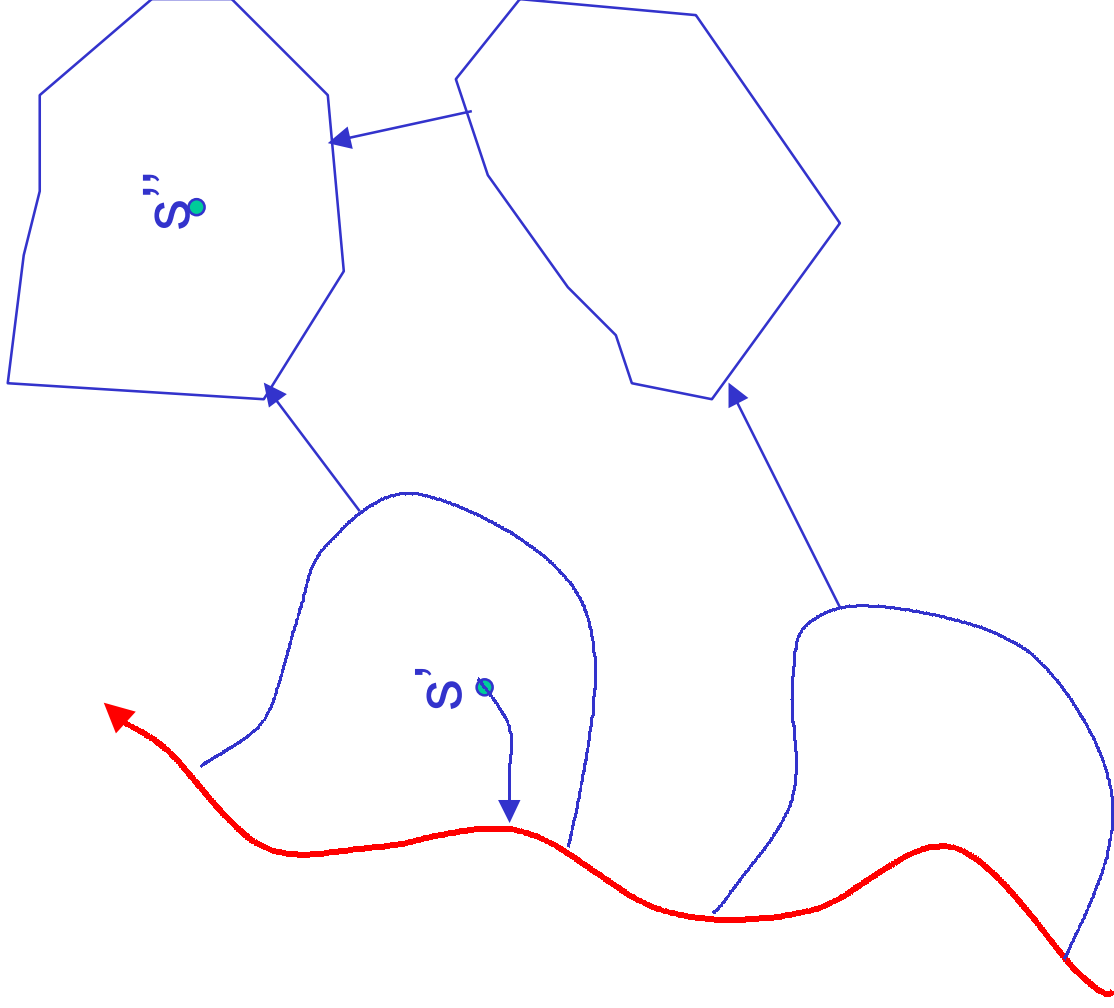
```
weiss := V; grau := schwarz := ∅;
CheckAU(s, ψ, χ)

weiss := weiss \ {s}; grau := grau ∪ {s};
IF L(s, A(ψ ∪ χ)) = F THEN EXIT CheckAU; END;
IF L(s, A(ψ ∪ χ)) = W THEN RETURN; END;
CTL(s, χ);
IF L(s, χ) = W THEN L(s, A(ψ ∪ χ)) := W; RETURN; END;
L(s, A(ψ ∪ χ)) := F;
CTL(s, ψ);
IF L(s, ψ) = F THEN EXIT CheckAU; END;
FOR ALL s': [s, s'] ∈ E DO
    IF s' ∈ weiss THEN
        CheckAU(s', ψ, χ);
    ELSIF s' ∈ grau THEN EXIT CheckAU; END;
END;
END;
grau := grau ∪ {s}; schwarz := schwarz ∪ {s}; L(s, A(ψ ∪ χ)) = W;
```

Verbesserung von CheckEU

1. Fall $s \neq E(\psi \cup \chi)$
 \Rightarrow alle s' im Suchraum: $s' \neq E(\psi \cup \chi)$
(sonst: $s \dots s' +$ Zeuge bei s' wäre Zeuge bei s !)
 2. Fall $s \models E(\psi \cup \chi) \rightarrow$ Stack zum Abbruchzeitpunkt bildet Zeuge
- 2a) s' auf dem Tiefensuchstack (“grau”)
 \rightarrow Stack ab s' ist immer noch Zeuge, also
 $s' \models E(\psi \cup \chi)$
- 2b) s' nicht mehr auf dem Tiefensuchstack (“schwarz”)
 \rightarrow (b.w.)

Verbesserung von CheckEU



$$s' \models E(\psi \cup \chi)$$

(s' in nicht beendeter SZK)

$$s'' \not\models E(\psi \cup \chi)$$

(s'' in beendeter SZK)
Argumentation analog
zu CheckAU

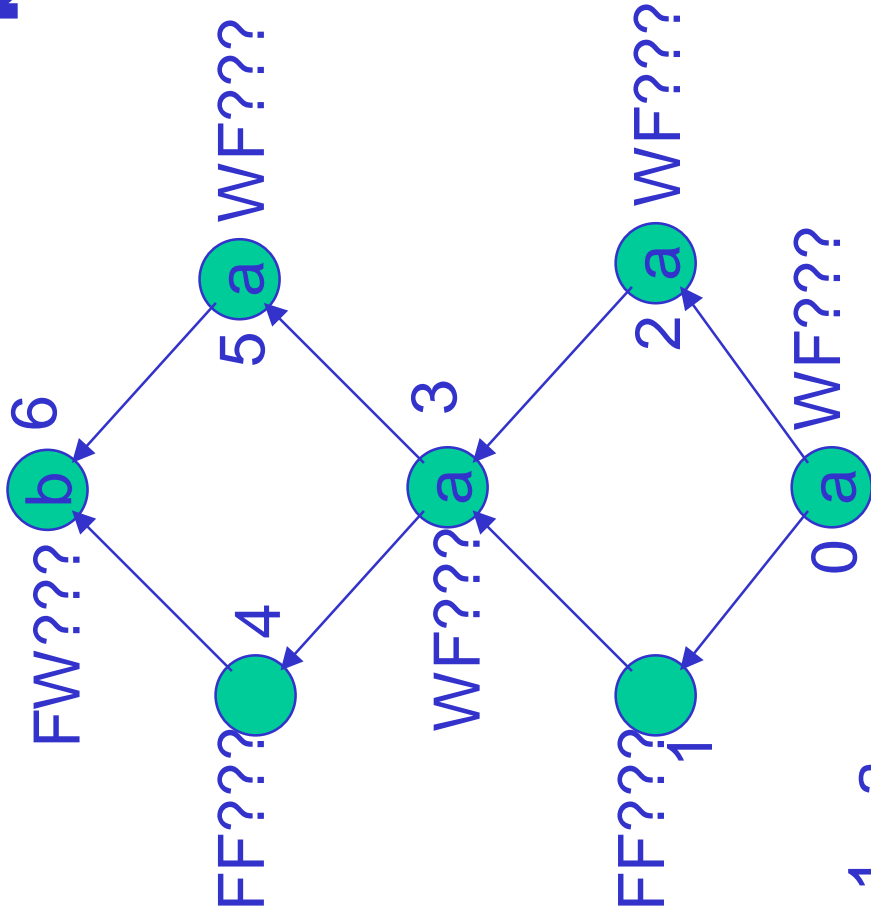
CheckEU - Pseudocode

```
maxdfs = 0; weiss := V; T := emptystack;

CheckEU(s,  $\psi$ ,  $\chi$ )
IF L(s, E( $\psi \cup \chi$ )) = W THEN EXIT CheckEU END;
IF L(s, E( $\psi \cup \chi$ )) = F THEN
    L(s, E( $\psi \cup \chi$ )) := F; RETURN
END;
CTL(s,  $\chi$ );
IFL(s,  $\chi$ ) = W THEN L(s, E( $\psi \cup \chi$ )) = W; EXIT; END
CTL(s,  $\psi$ );
IF L(s,  $\psi$ ) = F THEN L(s, E( $\psi \cup \chi$ )) = F; RETURN; END
L(s, E( $\psi \cup \chi$ )) := W; s.dfs = maxdfs; maxdfs += 1;
weiss := weiss \{s}; push(T, s); s.lowlink := s.dfs;
FOR s':[s, s'] ∈ E DO
    IF s' ∈ weiss THEN
        CheckEU(s',  $\psi$ ,  $\chi$ );
        s.lowlink := MIN(s.lowlink, s'.lowlink);
    ELSE
        IF s' ∈ T THEN s.lowlink := MIN(s.lowlink, s'.dfs); END
    END;
END
```

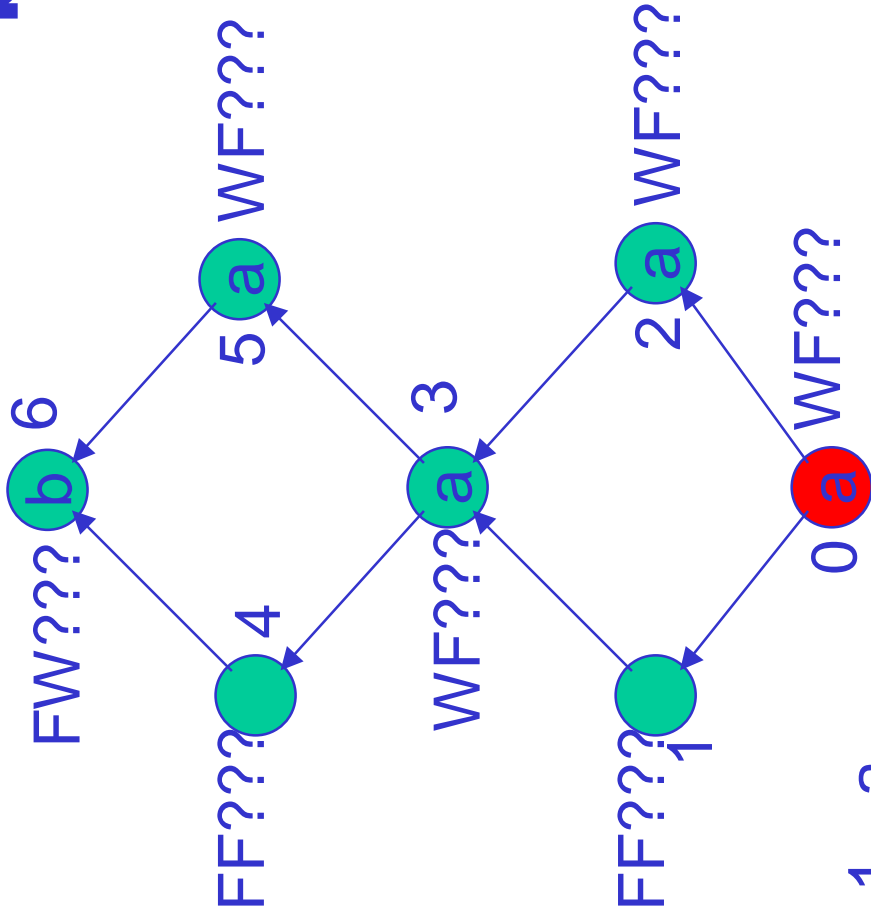
```
IF s.lowlink = s.dfs THEN
    REPEAT
        s' := pop(T);
        L(s', E( $\psi \cup \chi$ )) := F;
    UNTIL s = s'
END
```

Beispiel: $E(E(a \cup b) \cup A(a \cup b))$



1. a
2. b
3. $E(a \cup b)$
4. $A(a \cup b)$
5. $E(E(a \cup b) \cup A(a \cup b))$

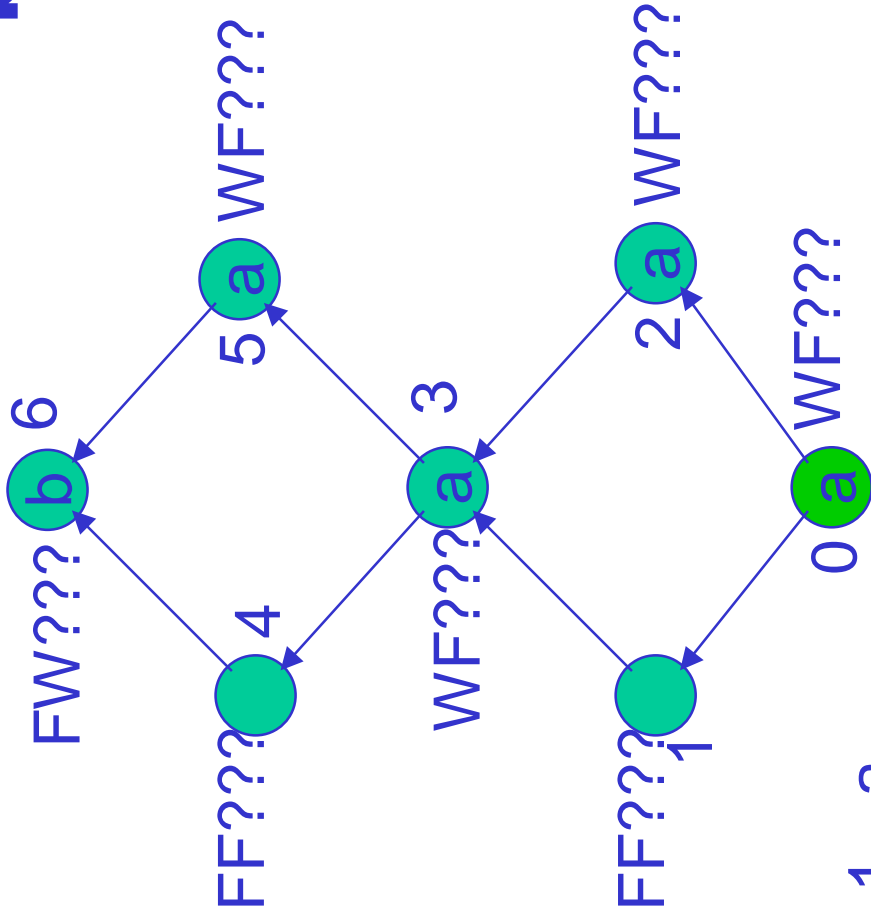
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0,E(a U b),A(a U b))

1. a
2. b
3. $E(a U b)$
4. $A(a U b)$
5. $E(E(a U b) U A(a U b))$

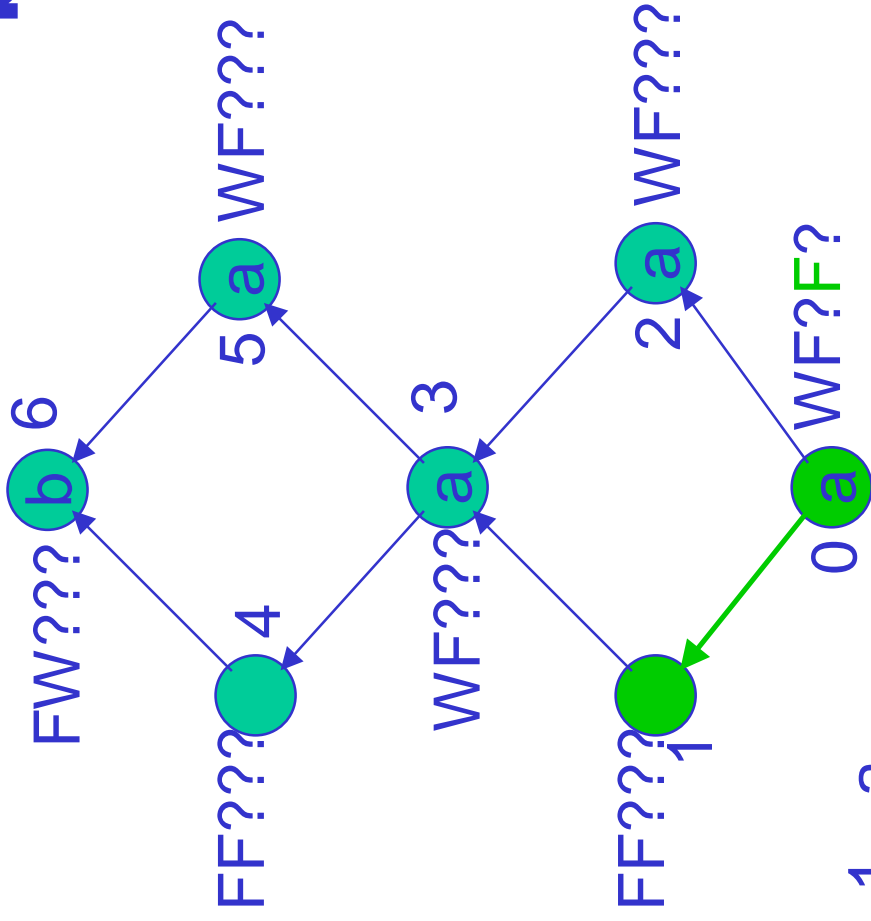
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0,E(a U b),A(a U b))
CheckAU(0,a,b)

1. a
2. b
3. $E(a U b)$
4. $A(a U b)$
5. $E(E(a U b) U A(a U b))$

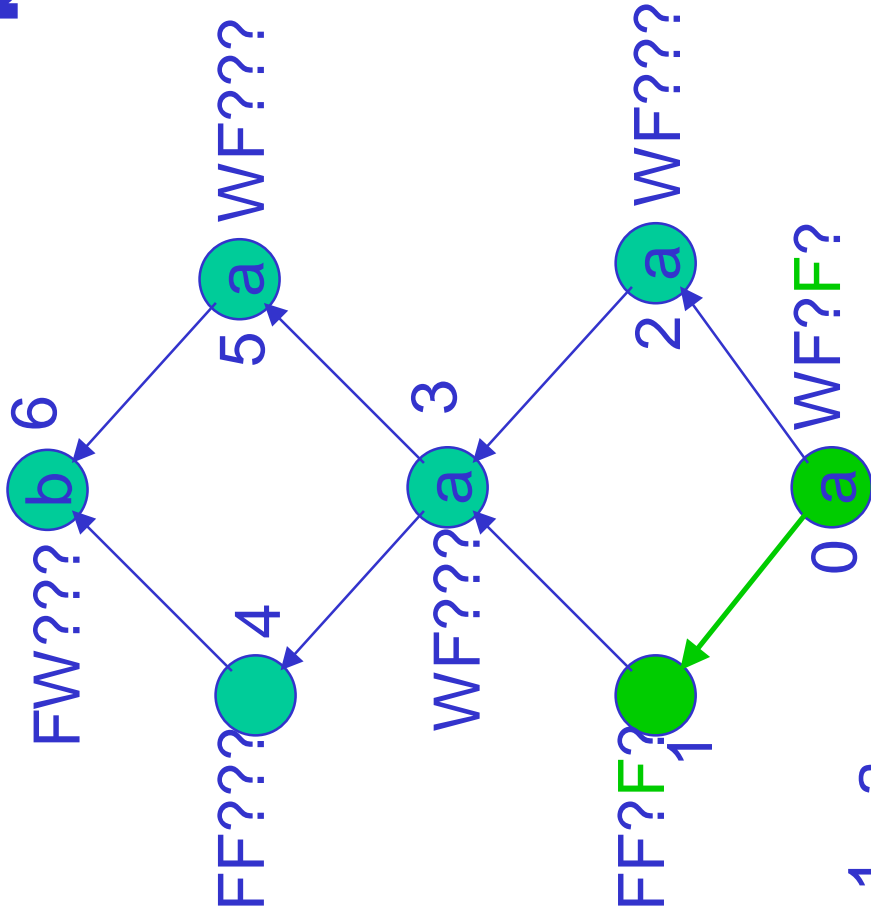
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0,E(a U b),A(a U b))
 CheckAU(0,a,b)

1. a
2. b
3. $E(a U b)$
4. $A(a U b)$
5. $E(E(a U b) U A(a U b))$

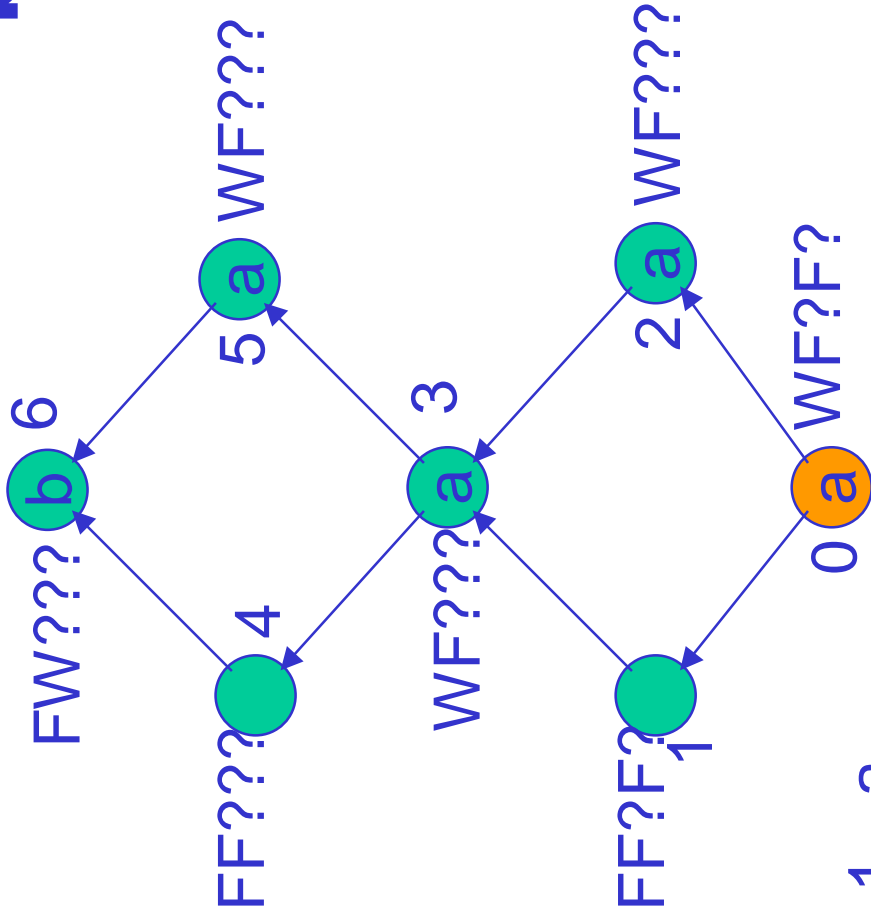
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0,E(a U b),A(a U b))
 CheckAU(0,a,b)

1. a
2. b
3. $E(a U b)$
4. $A(a U b)$
5. $E(E(a U b) U A(a U b))$

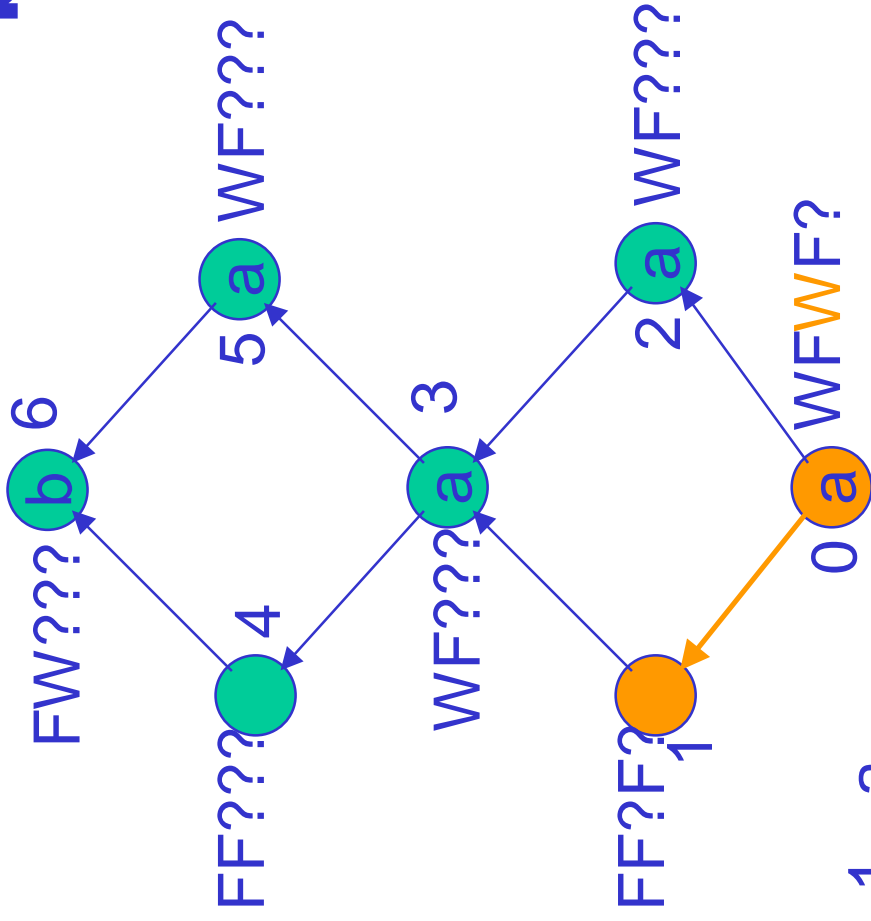
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0, E(a U b), A(a U b))
 CheckEU(0, a, b)

1. a
2. b
3. $E(a U b)$
4. $A(a U b)$
5. $E(E(a U b) U A(a U b))$

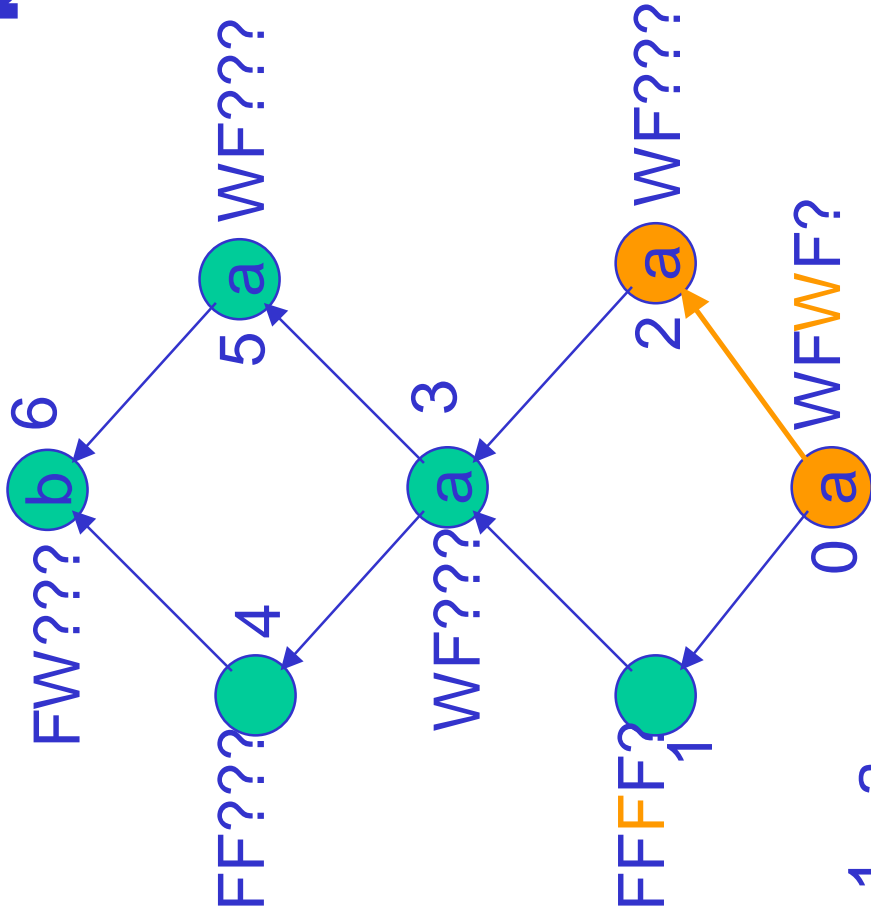
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0,E(a U b),A(a U b))
 CheckEU(0,a,b)

1. a
2. b
3. $E(a U b)$
4. $A(a U b)$
5. $E(E(a U b) U A(a U b))$

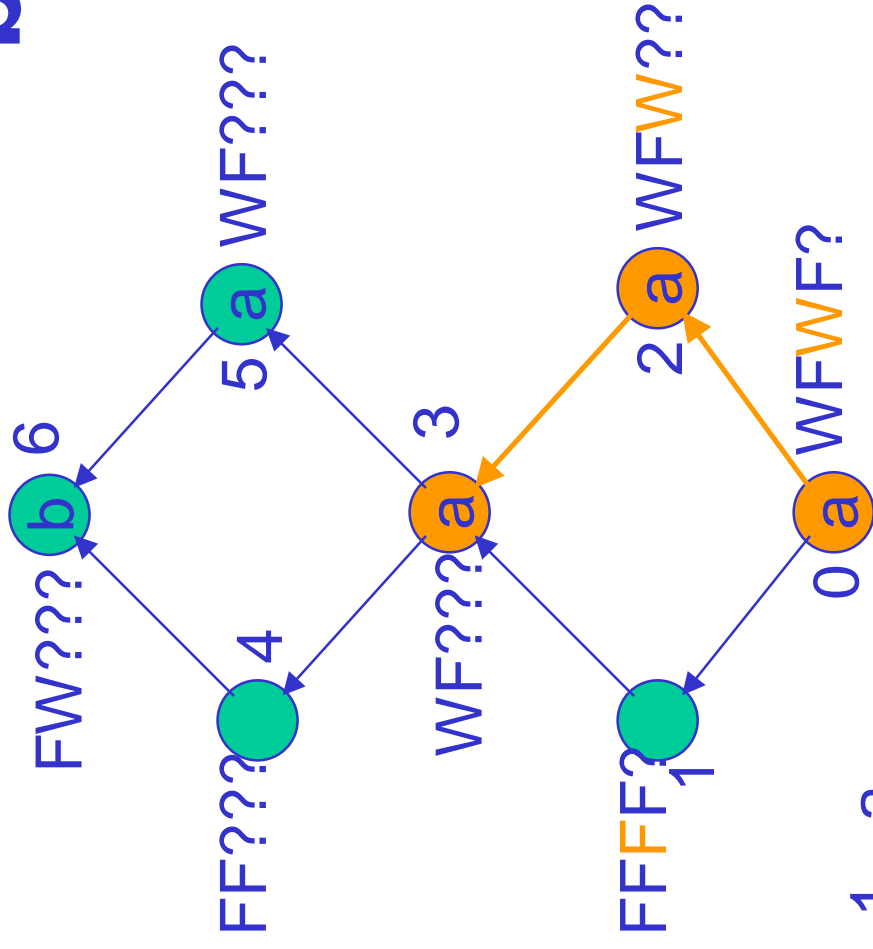
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0, E(a U b), A(a U b))
 CheckEU(0, a, b)

1. a
2. b
3. $E(a U b)$
4. $A(a U b)$
5. $E(E(a U b) U A(a U b))$

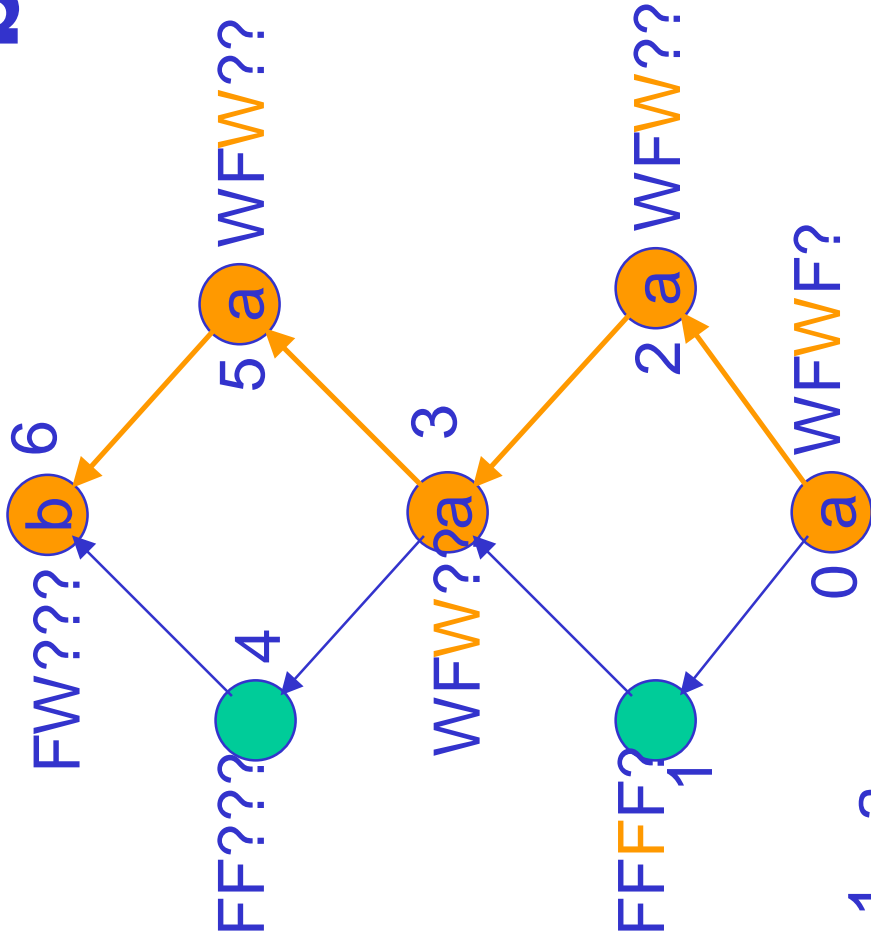
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0, E(a U b), A(a U b))
 CheckEU(0, a, b)

1. a
2. b
3. $E(a U b)$
4. $A(a U b)$
5. $E(E(a U b) U A(a U b))$

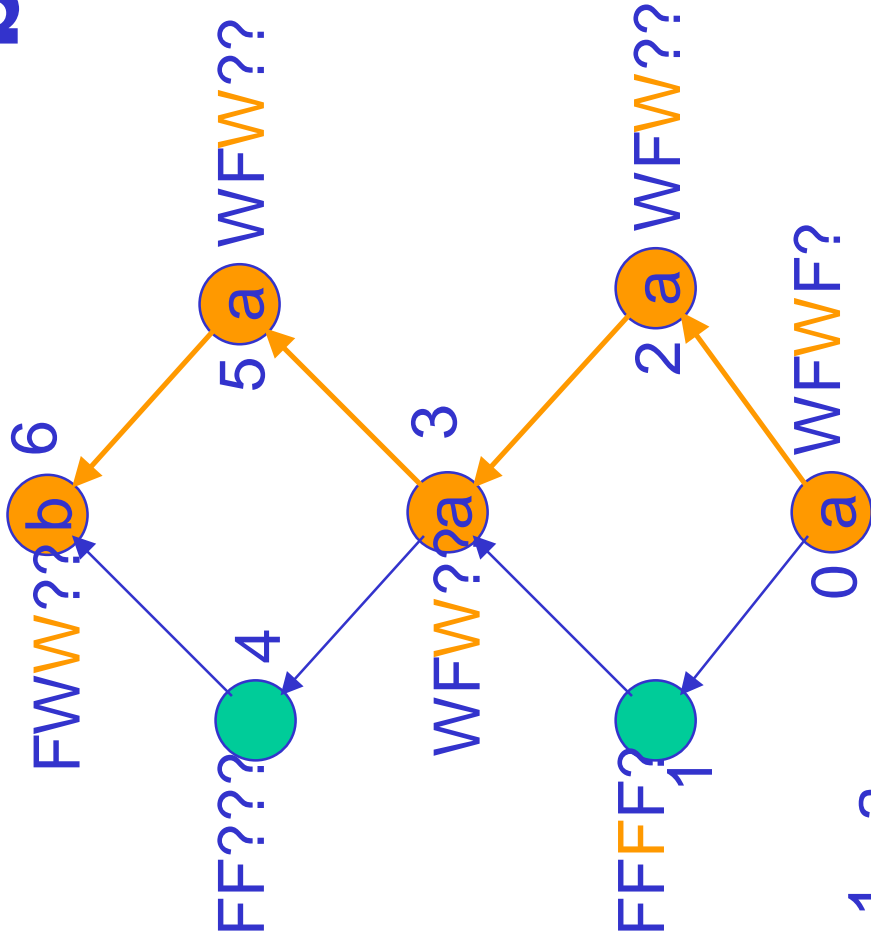
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0, E(a U b), A(a U b))
 CheckEU(0, a, b)

- 1. a
- 2. b
- 3. E(a U b)
- 4. A(a U b)
- 5. E(E(a U b) U A(a U b))

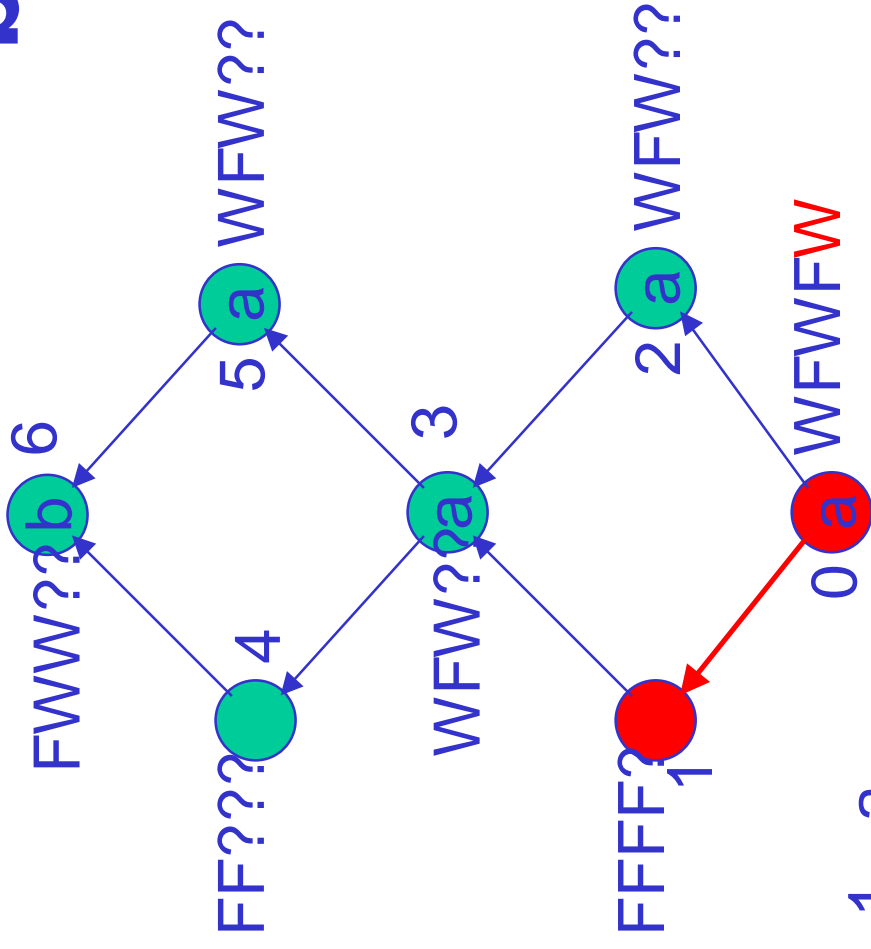
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0,E(a U b),A(a U b))
 CheckEU(0,a,b)

- 1. a
- 2. b
- 3. $E(a U b)$
- 4. $A(a U b)$
- 5. $E(E(a U b) U A(a U b))$

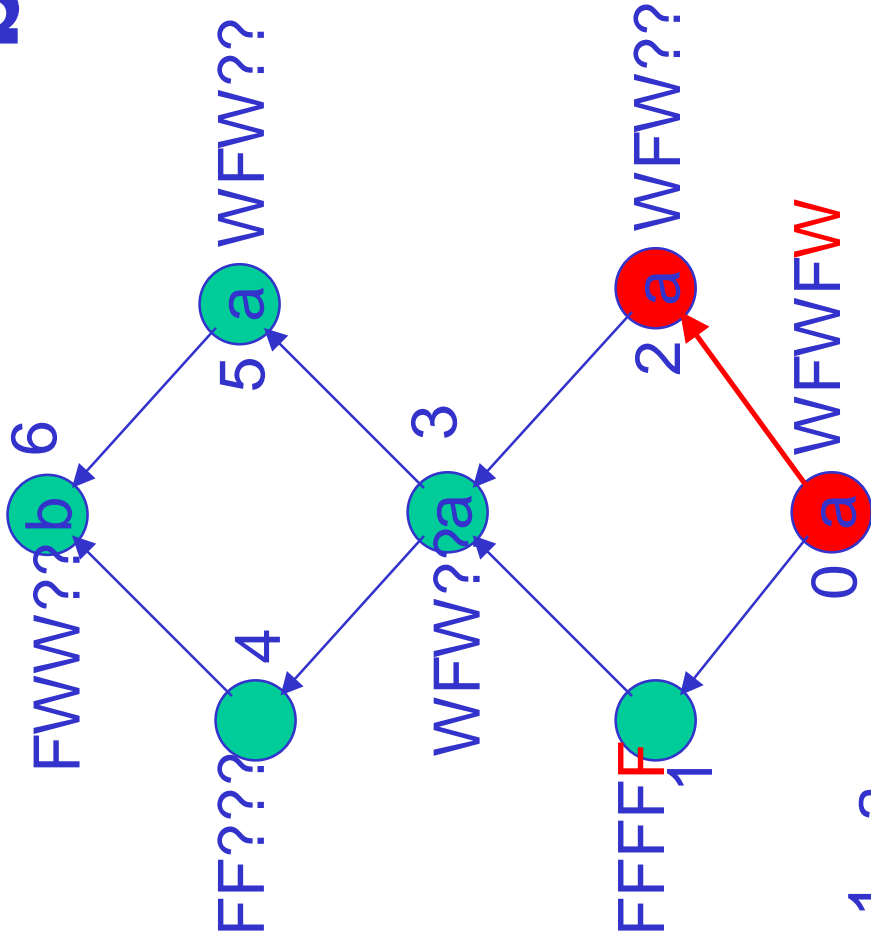
Beispiel: $E(E(a \cup b) \cup A(a \cup b))$



CheckEU(0, E(a U b), A(a U b))

1. a
2. b
3. $E(a \cup b)$
4. $A(a \cup b)$
5. $E(E(a \cup b) \cup A(a \cup b))$

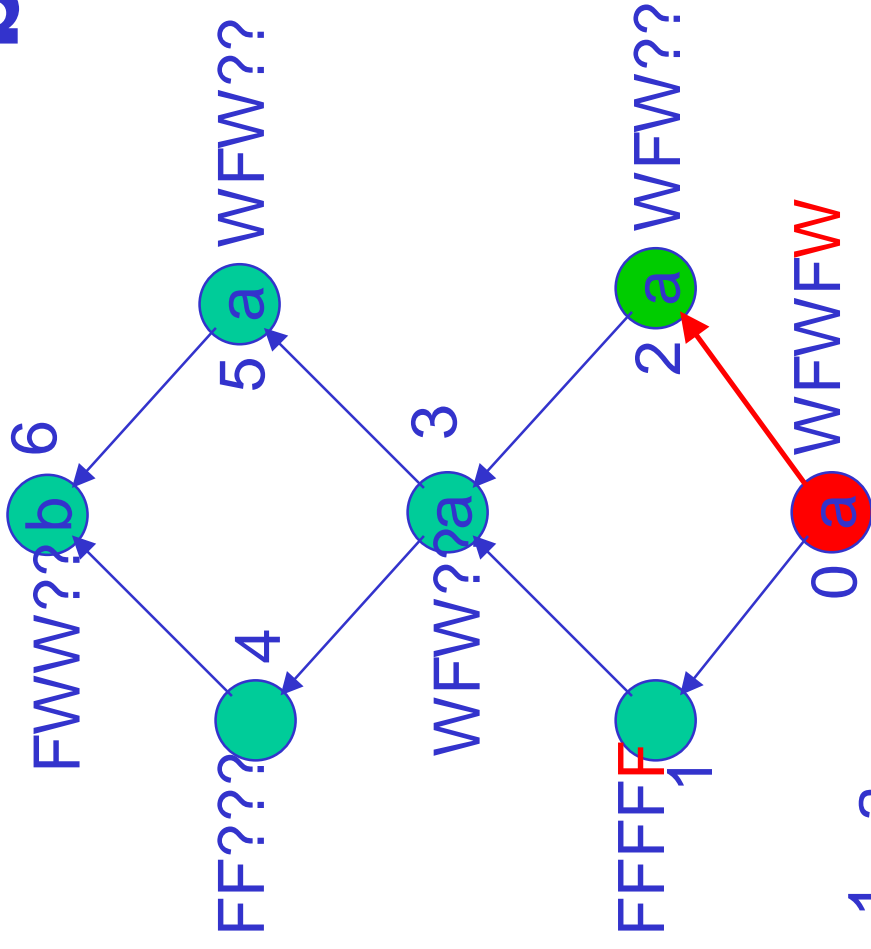
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0,E(a U b),A(a U b))

1. a
2. b
3. $E(a U b)$
4. $A(a U b)$
5. $E(E(a U b) U A(a U b))$

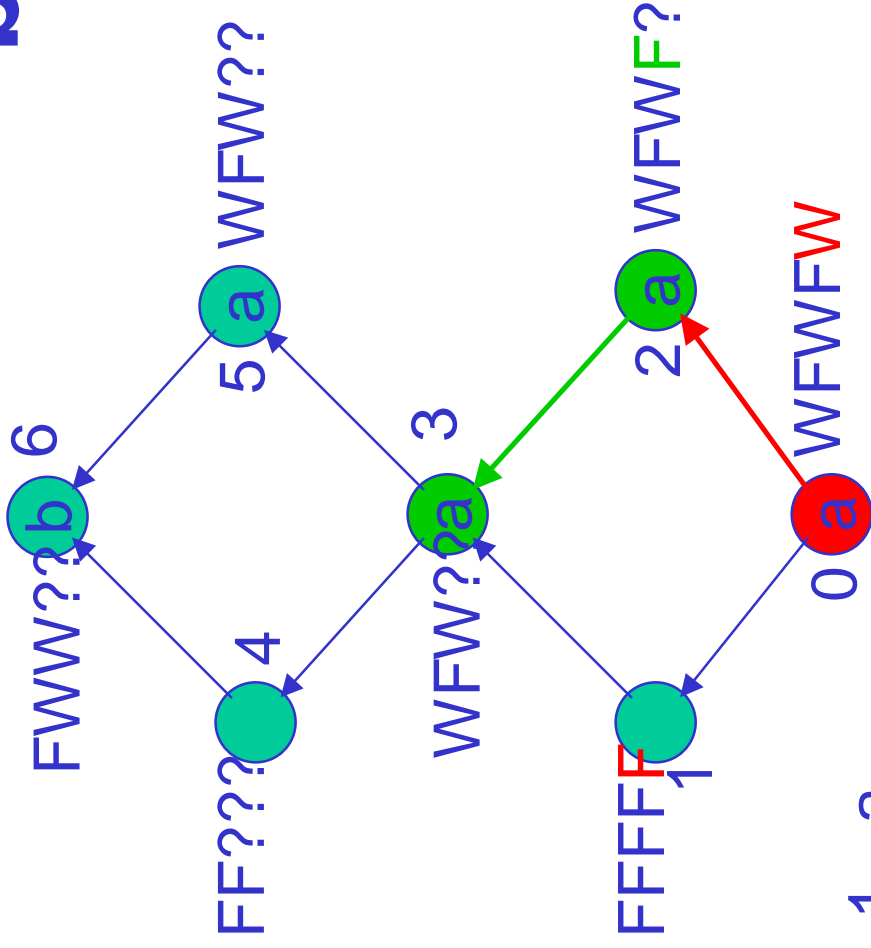
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0,E(a U b),A(a U b))
 CheckAU(1,a,b)

- 1. a
- 2. b
- 3. $E(a U b)$
- 4. $A(a U b)$
- 5. $E(E(a U b) U A(a U b))$

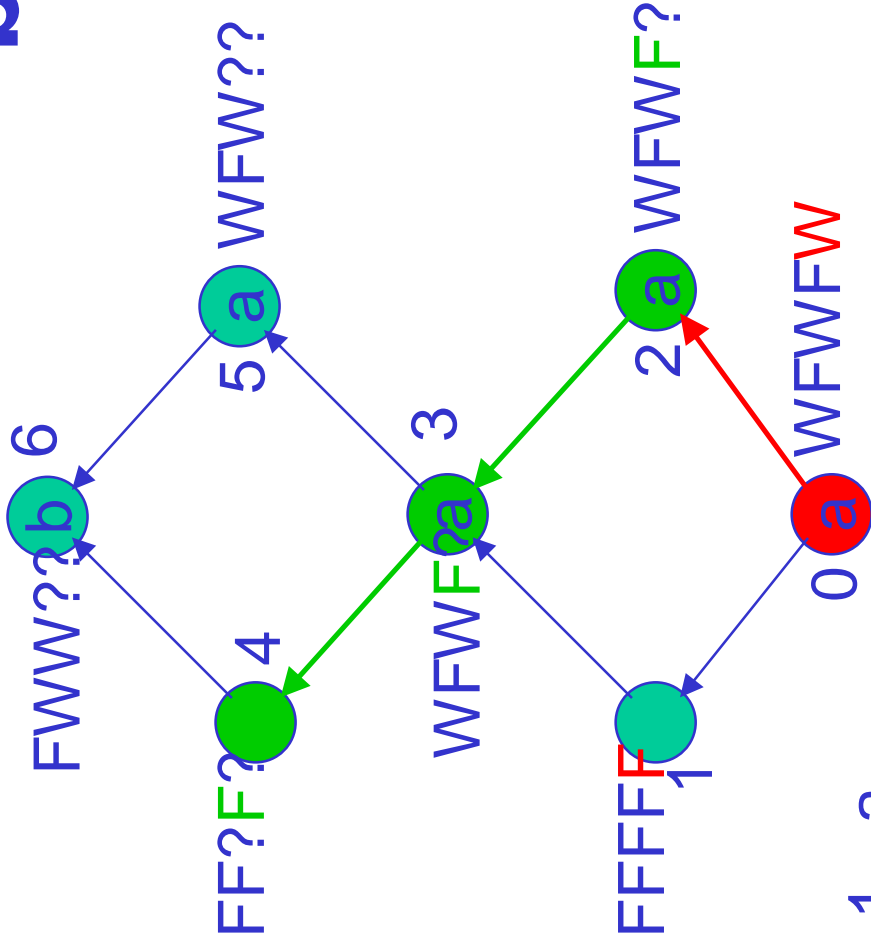
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0,E(a U b),A(a U b))
 CheckAU(1,a,b)

- 1. a
- 2. b
- 3. $E(a U b)$
- 4. $A(a U b)$
- 5. $E(E(a U b) U A(a U b))$

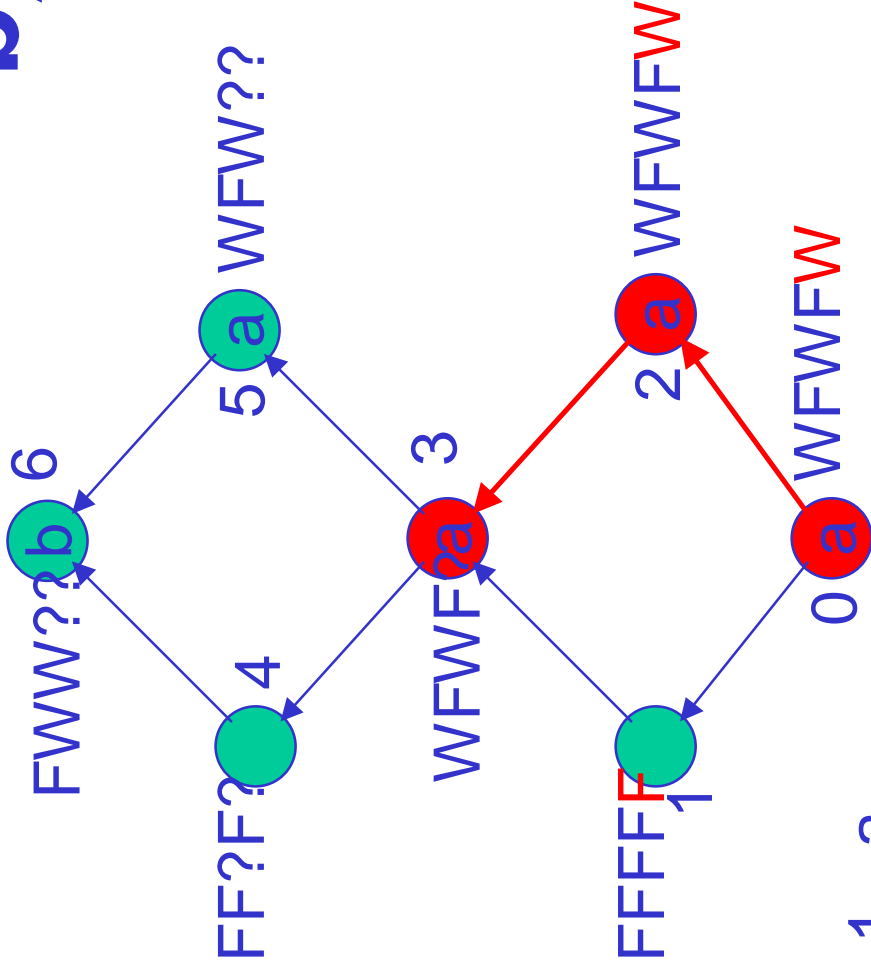
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0, E(a U b), A(a U b))
 CheckAU(1, a, b)

1. a
2. b
3. E(a U b)
4. A(a U b)
5. E(E(a U b) U A(a U b))

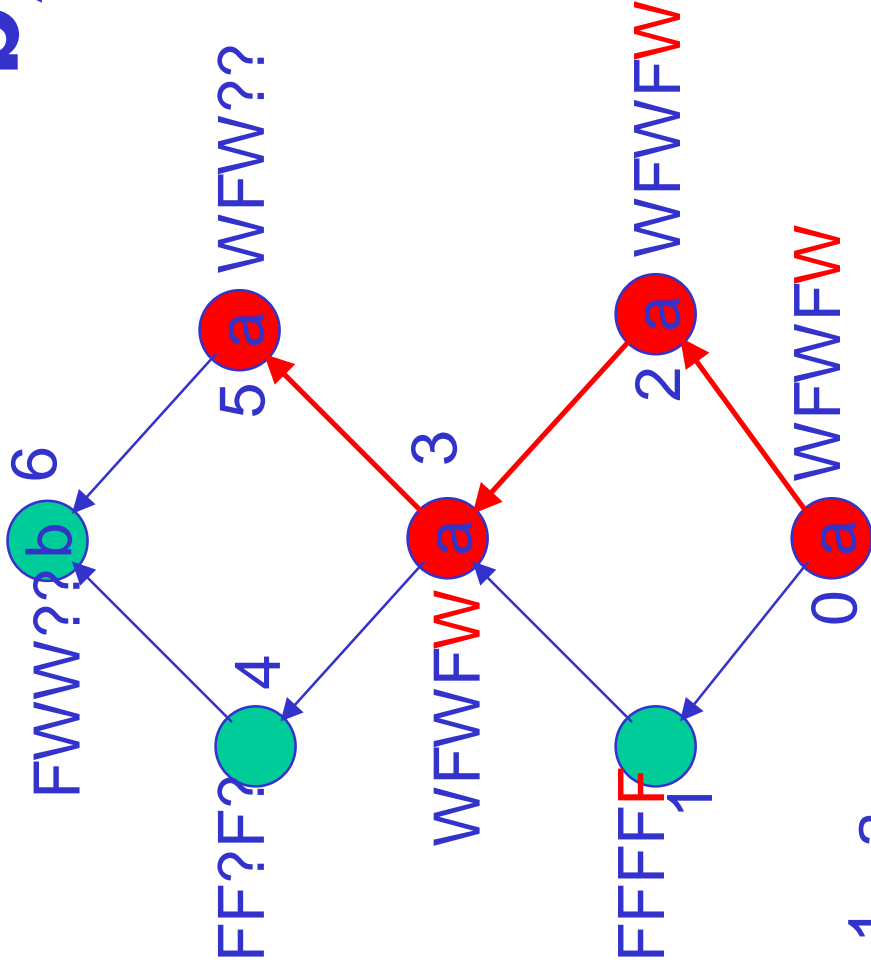
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0,E(a U b),A(a U b))

1. a
2. b
3. $E(a U b)$
4. $A(a U b)$
5. $E(E(a U b) U A(a U b))$

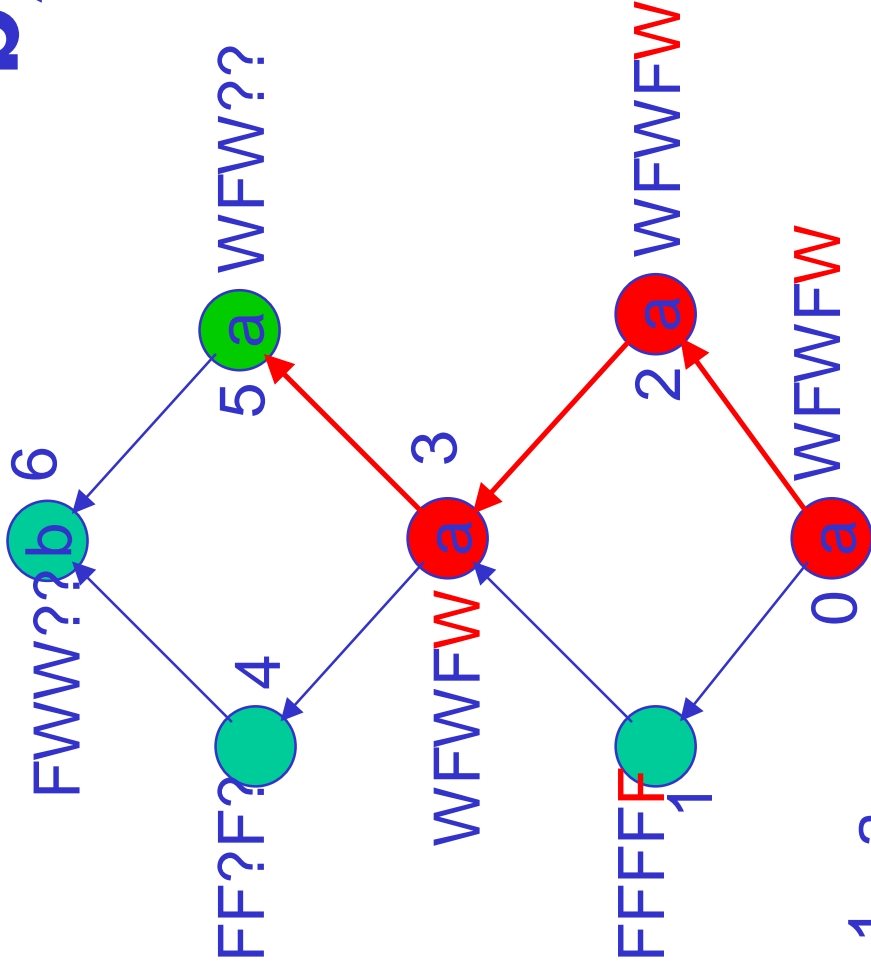
Beispiel: $E(E(a U b) U A(a U b))$



CheckEU(0, E(a U b), A(a U b))

1. a
2. b
3. $E(a U b)$
4. $A(a U b)$
5. $E(E(a U b) U A(a U b))$

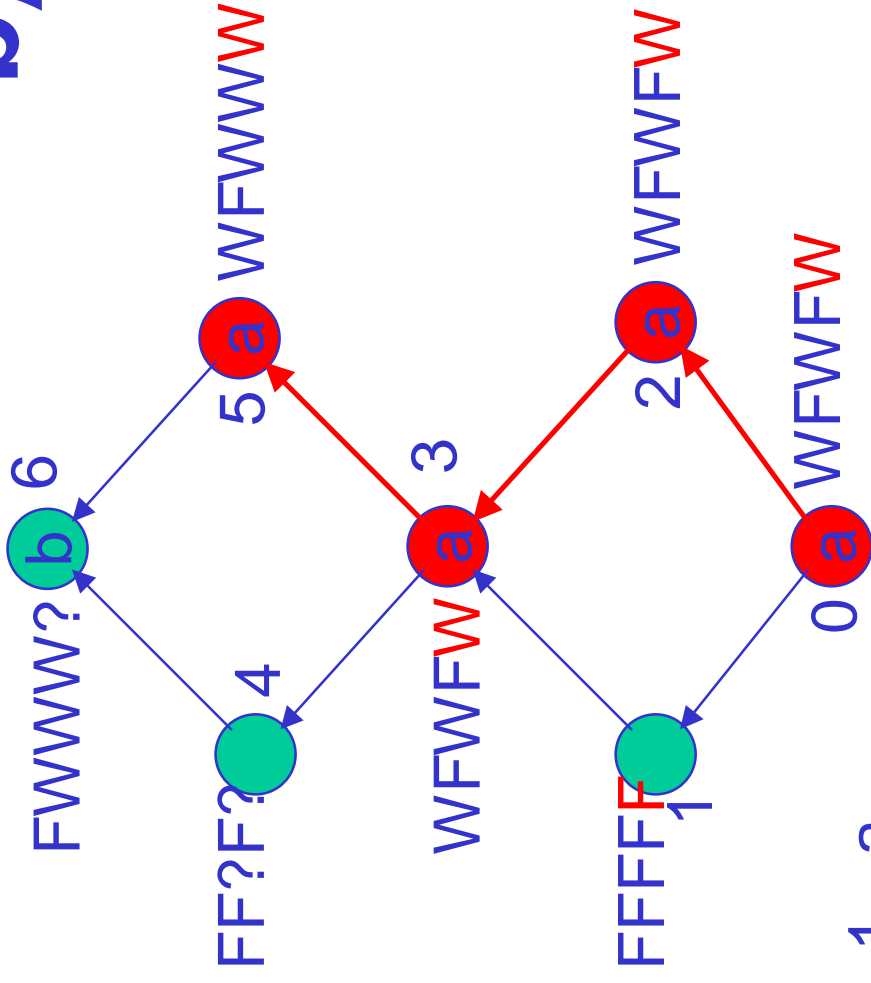
Beispiel: $E(E(a U b) U A(a U b))$



1. a
2. b
3. $E(a U b)$
4. $A(a U b)$
5. $E(E(a U b) U A(a U b))$

CheckEU(0, E(a U b), A(a U b))
 CheckAU(5, a, b)

Beispiel: $E(E(a U b) U A(a U b))$



1. a
2. b
3. $E(a U b)$
4. $A(a U b)$
5. $E(E(a U b) U A(a U b))$

CheckEU(0, E(a U b), A(a U b))

Zeugenpfad
gefunden!

fertig.

CTL Model Checking - Abschluß

Der Algorithmus ist Instanz der Technik
Dynamisches Programmieren

= mehrfach verwendete Zwischenresultate werden nur
einmal berechnet, bei wiederholter Verwendung nur
noch abgelesen.