KONGRUENZEN
VON VISIBLY PUSHDOWN SPRACHEN

REZART QELIBARI | PROSEMINAR WS14/15

INHALT

- Vorteile der VPLs / Warum der Aufwand?
- Definition: Visibly Pushdown Languages
- Charakterisierung durch Wortkongruenzen

WARUM DER AUFWAND?

AKTUELLE SITUATION

Situation:

Ziel u.a.: Wollen Programmflüsse überprüfen.

Modellierungen:

Wir habe reg. Sprachen und kontextfreie Sprachen

Problem:

- Reg. Sprachen sind nicht mächtig genug
- kontextfreie Sprachen sind nicht abgeschlossen

WARUM DER AUFWAND?

AKTUELLE SITUATION

Vorteile der VPLs /
Warum der Aufwand?
- **Aktuelle Situation**
- Einordnung der VPLs

Definition: Visibly Pushdown Automat

Charakterisierung durch Wortkongruenzen

Beispiel: Zur Anwendung des Mengendurchschnitts

L_1: mehr { als }

u_1={{{} v_1={} w_1={{{ }}

L_2: mehr } als {

u_2={ }}} v_2={} w_2={{ }}}

L_3: gleich viele { wie }

v_3={}

=> Visibly Pushdown Sprachen (VPLs)

WARUM DER AUFWAND?

EINORDNUNG DER VPLS

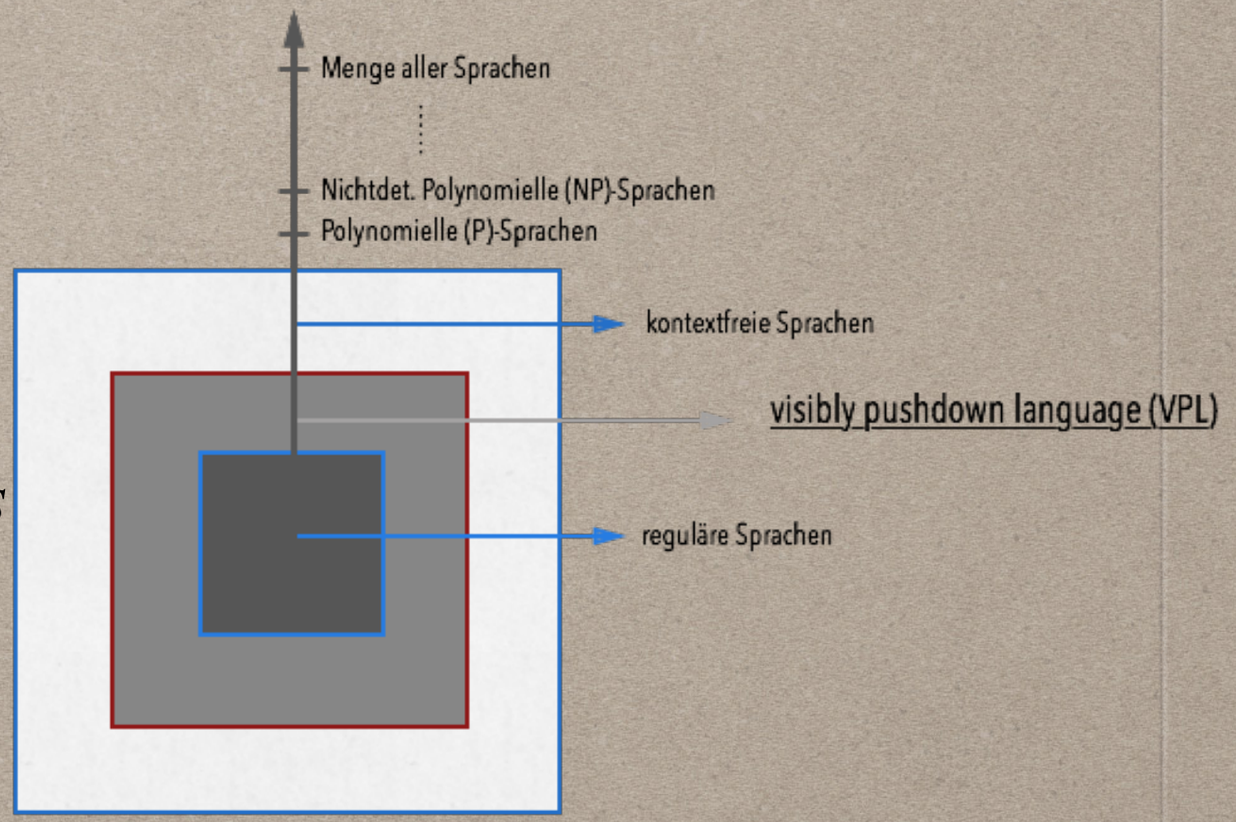
Vorteile der VPLs /
Warum der Aufwand?

- Aktuelle Situation
- Einordnung der VPLs

Definition: Visibly Pushdown Automat

Charakterisierung durch Wortkongruenzen

$$reg \subset VPLs \subset CFLs$$



VISIBLY PUSHDOWN AUTOMAT

VPLS UND VPAS

VPLs werden von sog. Visibly Pushdown Automaten erkannt.

- Regulärer Automat erweitert um einen Stack, wobei das Eingabewort die erlaubten Stackaktionen (push,pop,nop) definiert.

- Deterministische und nichtdeterministische Automaten sind hier - im Gegensatz zu den PushDownAutomaten - gleich mächtig.
=> Zur Vereinfachung beschränken wir uns auf deterministische Automaten.

VISIBLY PUSHDOWN AUTOMAT

VPA DEFINITION

Vorteile der VPLs /
Warum der Aufwand?

Definition: Visibly Pushdown Automat

- VPLs und VPAs
- VPA Definition
- Beispiel für eine VPL/VPA

Charakterisierung durch Wortkongruenzen

Eingabealphabet:

Ein Pushdown Alphabet ist ein Tripel:

$\hat{\Sigma} = (\Sigma_{call}, \Sigma_{ret}, \Sigma_{\text{int}})$

Hierbei sind $\Sigma_{call}, \Sigma_{ret}$ und Σ_{int} endlich und paarweise disjunkte Menge.

Def: VPA

Ein Visibly Pushdown Automat auf endl. Wörtern über Σ^* bei Beachtung von $\hat{\Sigma}$ ist ein Tupel $M = (Q, q_0, \Gamma, \delta, Q_F)$:

Q ist ein endl. Menge von Zuständen
$q_0 \in Q$ ist der Startzustand
Γ ist das Stackalphabet mit Bottom-of-Stack Symbol: $\perp$
δ ist die Transitionsfunktion
$Q_F \subseteq Q$ ist die Menge der Endzustände

Kommentar:

Das Tripel ist eine Einteilung (Partition) des Eingabealphabets Σ. Dadurch können alle Stackaktionen (push, pop, nop) vom Eingabewort gesteuert werden.

Bottom-of-Stack: Dieses Symbol verhindert einen leeren Stack (Siehe Akzeptanz)

VISIBLY PUSHDOWN AUTOMAT

VPA DEFINITION

Vorteile der VPLs /
Warum der Aufwand?

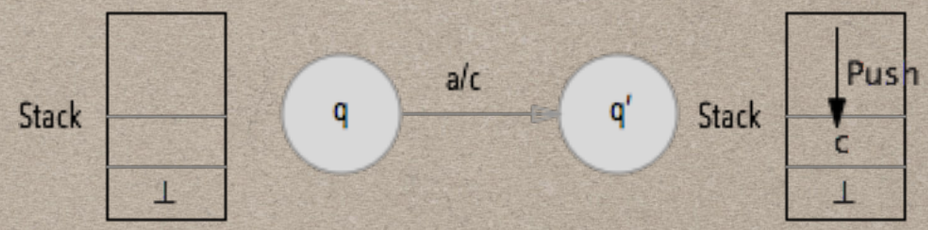
Definition: Visibly Pushdown Automat

- VPLs und VPAs
- VPA Definition
- Beispiel für eine VPL/VPA

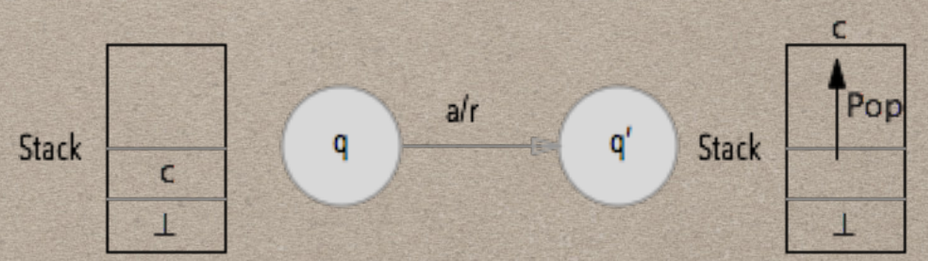
Charakterisierung durch Wortkongruenzen

Transitionen:

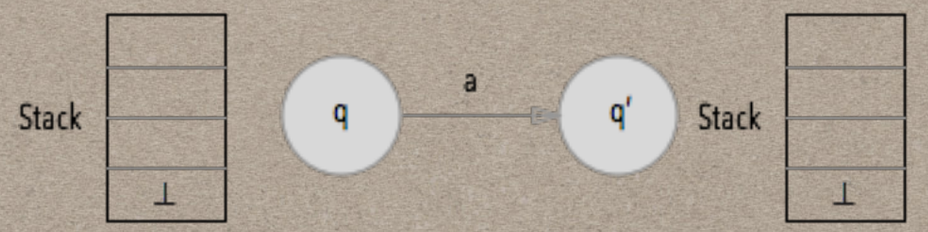
- Call-Transitionen:



- Return-Transitionen:



- Internal-Transitionen (Lokale Aktion)



VISIBLY PUSHDOWN AUTOMAT

BEISPIEL FÜR EINE VPL/VPA

Vorteile der VPLs /
Warum der Aufwand?

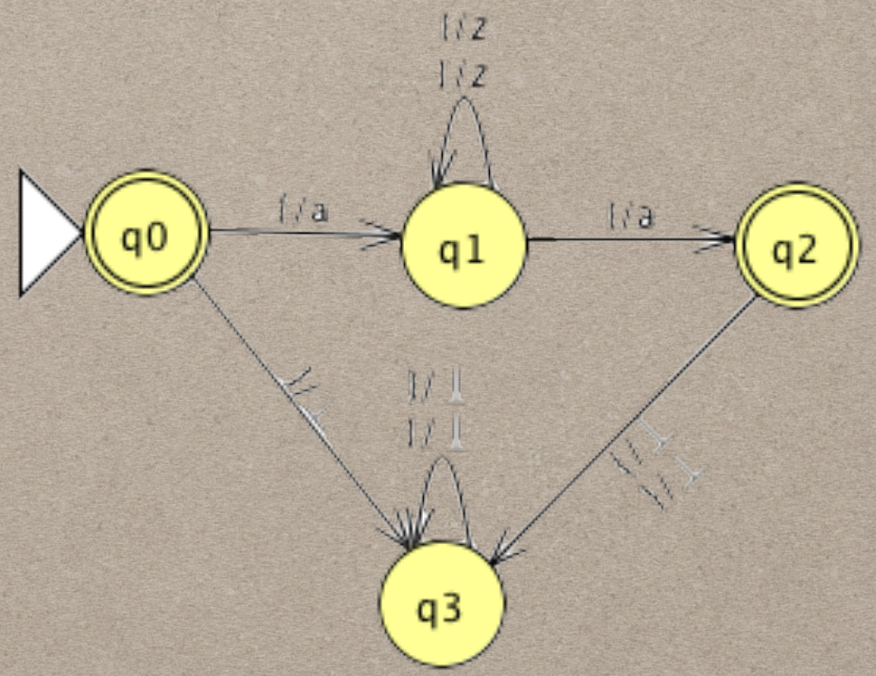
Definition: Visibly Pushdown Automat

- VPLs und VPAs
- VPA Definition
- Beispiel für eine VPL/VPA

Charakterisierung durch Wortkongruenzen

Betrachte:

$$L = \{(^n\ )^n \mid n \geq 0\}$$



CHARAKTERISIERUNG DURCH KONGRUENZEN

BEGRIFFSKLÄRUNG

Vorteile der VPLs /
Warum der Aufwand?

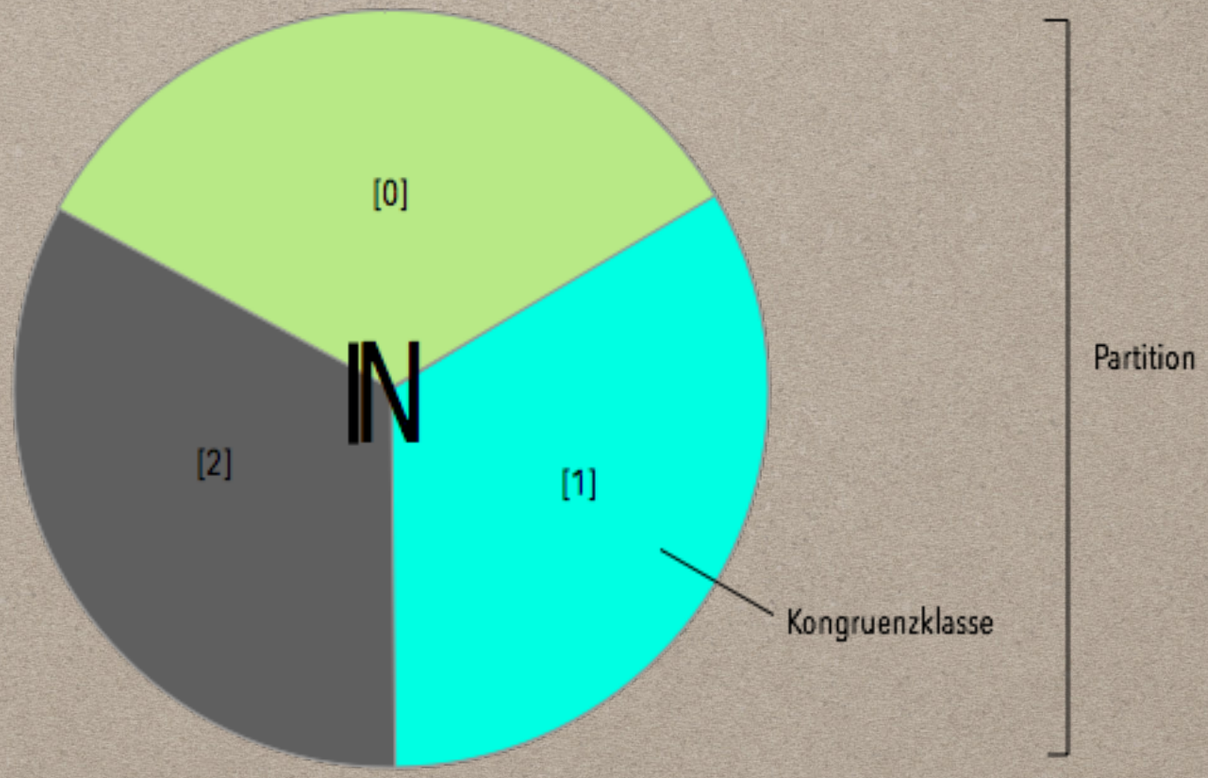
Definition: Visibly Pushdown Automat

Charakterisierung durch Wortkongruenzen

- **Begriffsklärung**
- Nerode-Kongruenz
- VPL-Kongruenz

Kongruenz: (am Beispiel des Modulo Operators)

Intuitiv: Operator, der Elemente einer Menge zusammenfasst.



Hier: Modulo 3 Kongruenzklassen

CHARAKTERISIERUNG DURCH KONGRUENZEN

NERODE-KONGRUENZ

Vorteile der VPLs /
Warum der Aufwand?

Definition: Visibly Pushdown Automat

Charakterisierung durch Wortkongruenzen

- Begriffsklärung
- **Nerode-Kongruenz**
- VPL-Kongruenzen

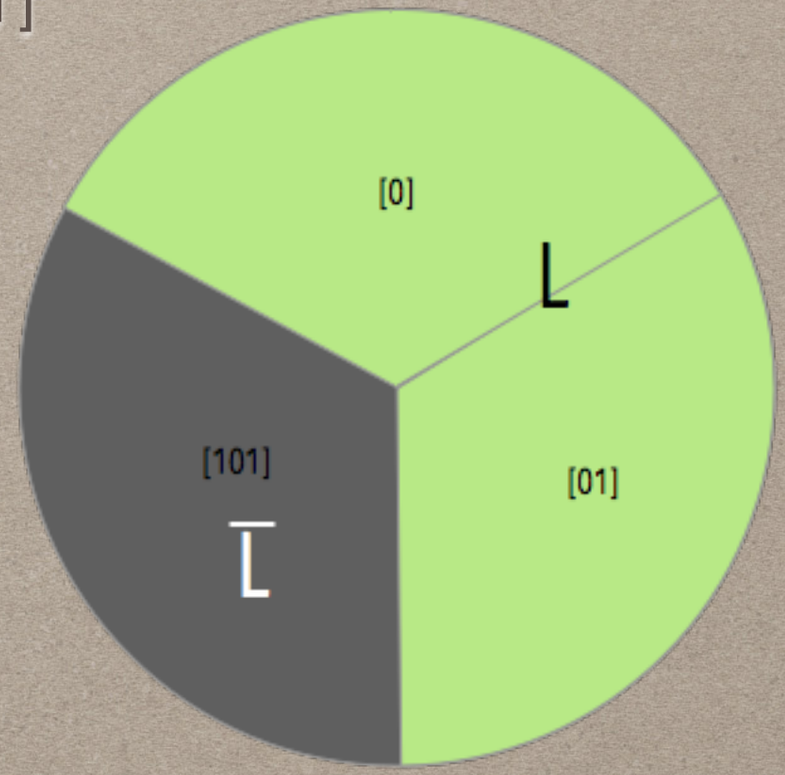
Wir wissen:

$$u \equiv_L v \Leftrightarrow \forall w \in \Sigma^* : uw \in L \Leftrightarrow vw \in L$$

Beispiel:

Gegeben L={w | w = 0*1* }, dann:

1. [ε] = [0] = [0...0]
2. [1] = [01] = [0...01]= [0...01...1]
3. [10] = [0..010] = [0...01...10]



CHARAKTERISIERUNG DURCH KONGRUENZEN

VPL-KONGRUENZEN

Können wir die Nerode-Kongruenz übernehmen?

Sei das Alphabet

$$\Sigma_{call}=\{ (\} , \quad \Sigma_{ret}=\{) \}, \quad \Sigma_{int}=\{\}$$

gegeben.
Betrachte:

$$L=\{s \mid s \text{ ist well-matched}\}$$

Dazu zwei Wörter:

u=(( )) v=()

=> Es gilt: u und v sind in L

Erinnerung: $u \equiv_L v \Leftrightarrow \forall w \in \Sigma^* : uw \in L \Leftrightarrow vw \in L$

u.(= (( )).(=> ist nicht in L
u.) = (( )).) => ist nicht in L
für v analog.
=> Alle Wörter sind in nur einer Kongruenzklasse. Also Nein!

CHARAKTERISIERUNG DURCH KONGRUENZEN

VPL-KONGRUENZEN

Im folgenden sind diese Mengen wichtig:

Die Menge MatchedCall: Menge aller Wörter bei der jedem Call Symbol ein Return Symbol folgt (muss nicht direkt folgen).

Die Menge MatchedReturn: Menge aller Wörter bei der jedem Return Symbol ein Call Symbol vorrausgeht (muss nicht direkt davor sein).

Die Menge WellMatched: Menge aller Wörter bei der jedem Call Symbol ein Return Symbol folgt und jedem Return Symbol ein Call Symbol vorausgeht (muss auch nicht direkt sein)

CHARAKTERISIERUNG DURCH KONGRUENZEN

VPL-KONGRUENZEN

Eigenschaften der Kongruenz/
Der Unterschied zu regulären Sprachen

Das Eingabewort kann Stackaktionen auslösen.

Unterscheide deshalb 4. Fälle:

1. Das Eingabewort u enthält keine Calls und Returns.
   Dann sollte die Kongruenz gleich der Nerode-Kongruenz sein.
2. Das Eingabewort enthält gleich viele Calls und Returns (Well-Matched).
3. Das Eingabewort enthält mehr Calls als Returns.
4. Das Eingabewort enthält mehr Returns als Calls.

CHARAKTERISIERUNG DURCH KONGRUENZEN

VPL-KONGRUENZEN

Betrachte Punkt 2:

Das Eingabewort enthält gleich viele Calls wie Returns.

=> Diese Wörter müssen von beiden Seiten erweitert werden, damit sie noch in der Sprache liegen

Definiere deshalb:

$$w_1 \approx w_2 \Leftrightarrow \forall u,v \in \Sigma^* : uw_1v \in L \Leftrightarrow uw_2v \in L$$

Beispiel: $L = \{(^n)^n \mid n \geq 0\}$

Dann gibt es nur zwei Klassen: $\{(^n)^n \mid n \geq 0\}$ und die Komplement Menge $\Sigma^* \setminus L$

CHARAKTERISIERUNG DURCH KONGRUENZEN

VPL-KONGRUENZEN

Punkt 3:

Das Eingabewort enthält mehr Calls als Returns.

=> Hier enthält der Stack mehrere Symbole, die nicht durch Pop Transitionen entfernt werden. Intuition: Solche Wörter können intuitiv nicht von Wörtern unterschieden werden, die den Stack nicht inspizieren.

Definiere deshalb:

$$u_1, u_2 \in \Sigma^*, u_1 \equiv u_2 \Leftrightarrow \forall v \in MR(\hat{\Sigma}) : u_1 v \in L \Leftrightarrow u_2 v \in L$$

Beispiel: $L = \{(^n)^n \mid n \geq 0\}$

Dann gibt es wieder nur eine Kongruenzklasse

CHARAKTERISIERUNG DURCH KONGRUENZEN

VPL-KONGRUENZEN

Punkt 4:

Das Eingabewort enthält mehr Returns als Calls.

=> Die Returns, die keinen korrespondierenden Call haben, arbeiten wie internal Transitionen, da das Bottom-of-Stack Symbol nur gelesen, aber nicht gepoppt werden kann.

Definiere deshalb:

$$u_1, u_2 \in MC(\hat{\Sigma}), u_1 \sim_0 u_2 \Leftrightarrow \forall v \in \Sigma^* : u_1 v \in L \Leftrightarrow u_2 v \in L$$

Das stimmt mit der Definition der Nerode-Kongruenz überein.

Beispiel: $L = \{(^n)^n \mid n \geq 0\}$

Dann gibt es wieder nur eine Kongruenzklasse.

CHARAKTERISIERUNG DURCH KONGRUENZEN

VERGLEICH

Vorteile der VPLs /
Warum der Aufwand?

Definition: Visibly Pushdown Automat

Charakterisierung durch Wortkongruenzen
- Begriffsklärung
- Nerode-Kongruenz
- VPL-Kongruenzen

1. $w_1, w_2 \in MC(\hat{\Sigma}), w_1 \approx w_2 \Leftrightarrow \forall u, v \in \Sigma^* : uw_1v \in L \Leftrightarrow uw_2v \in L$

2. $u_1, u_2 \in \Sigma^*, u_1 \equiv u_2 \Leftrightarrow \forall v \in MR(\hat{\Sigma}) : u_1v \in L \Leftrightarrow u_2v \in L$

3. $u_1, u_2 \in MC(\hat{\Sigma}), u_1 \sim_0 u_2 \Leftrightarrow \forall v \in \Sigma^* : u_1v \in L \Leftrightarrow u_2v \in L$

ENDE

Quellen:

[1]: Congruences for visibly pushdown automata, Alur et. al, PDF

[2]: Visibly pushdown languages, Alur et. al, PDF