A Pumping Lemma for and Closure Properties of Context-Free Languages

Martin Fränzle

Informatics and Mathematical Modelling

The Technical University of Denmark

What you'll learn

1. **A pumping lemma for CFLs:**
   - The lemma
   - Its use in proving languages non-CFL
   - $\leadsto$ Insight in the limits of CFLs

2. **Closure properties of CFLs:**
   - The operation of substitution — generalizing homomorphisms
   - $\leadsto$ Closure under union, concatenation, Kleene closure, homomorphism
   - Non-closure under intersection, complement, difference
   - Closure under intersection with a *regular* language
   - Closure under inverse homomorphism

A pumping lemma for CFLs

Rationale

- As CFGs build strings of arbitrary length – if the language defined is not finite — from a finite set of rules, it is reasonable to expect that all sufficiently long words of the language have some kind of repeatable pattern.

- Obviously, this pattern can be more complex than that occurring in regular languages (why?)

Rationale

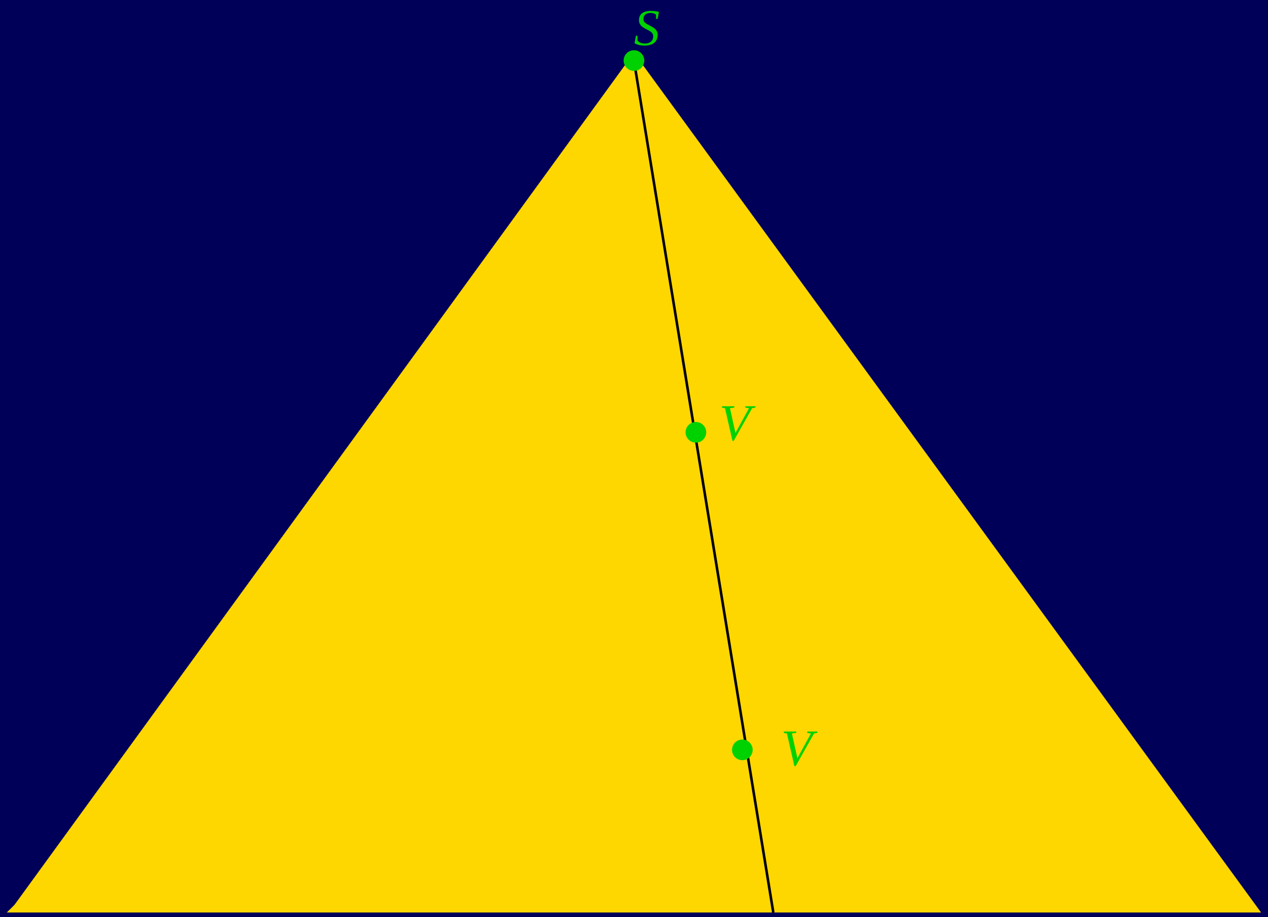
- As CFGs build strings of arbitrary length – if the language defined is not finite — from a finite set of rules, it is reasonable to expect that all sufficiently long words of the language have some kind of repeatable pattern.
- Obviously, this pattern can be more complex than that occurring in regular languages (why?)

Program:

1. Identify the kind of pattern;
2. derive a method for proving languages non-contextfree.

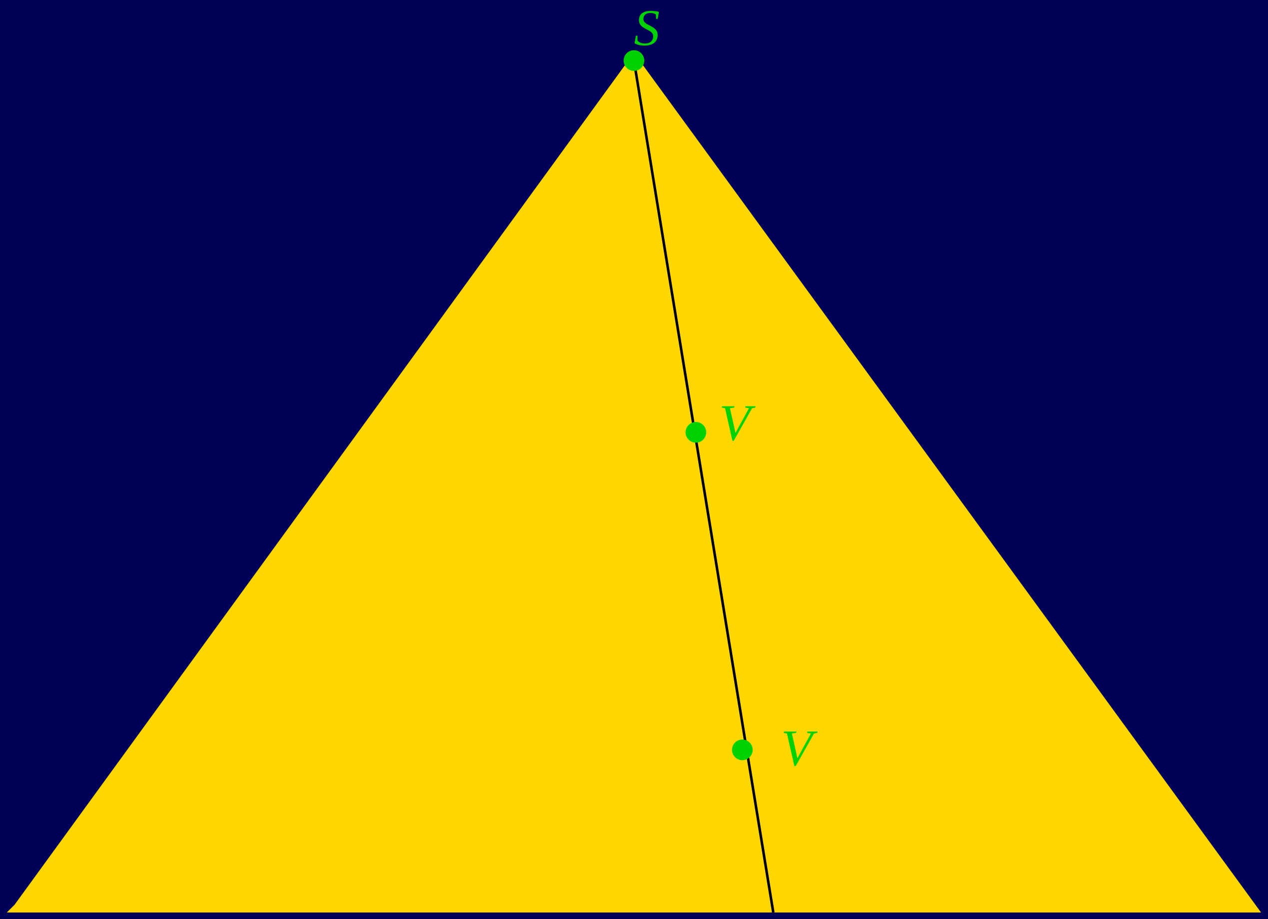
Idea

Any parse-tree deeper than the number of variables in the CFG must have some variable occur at least twice on its deepest path:



Idea

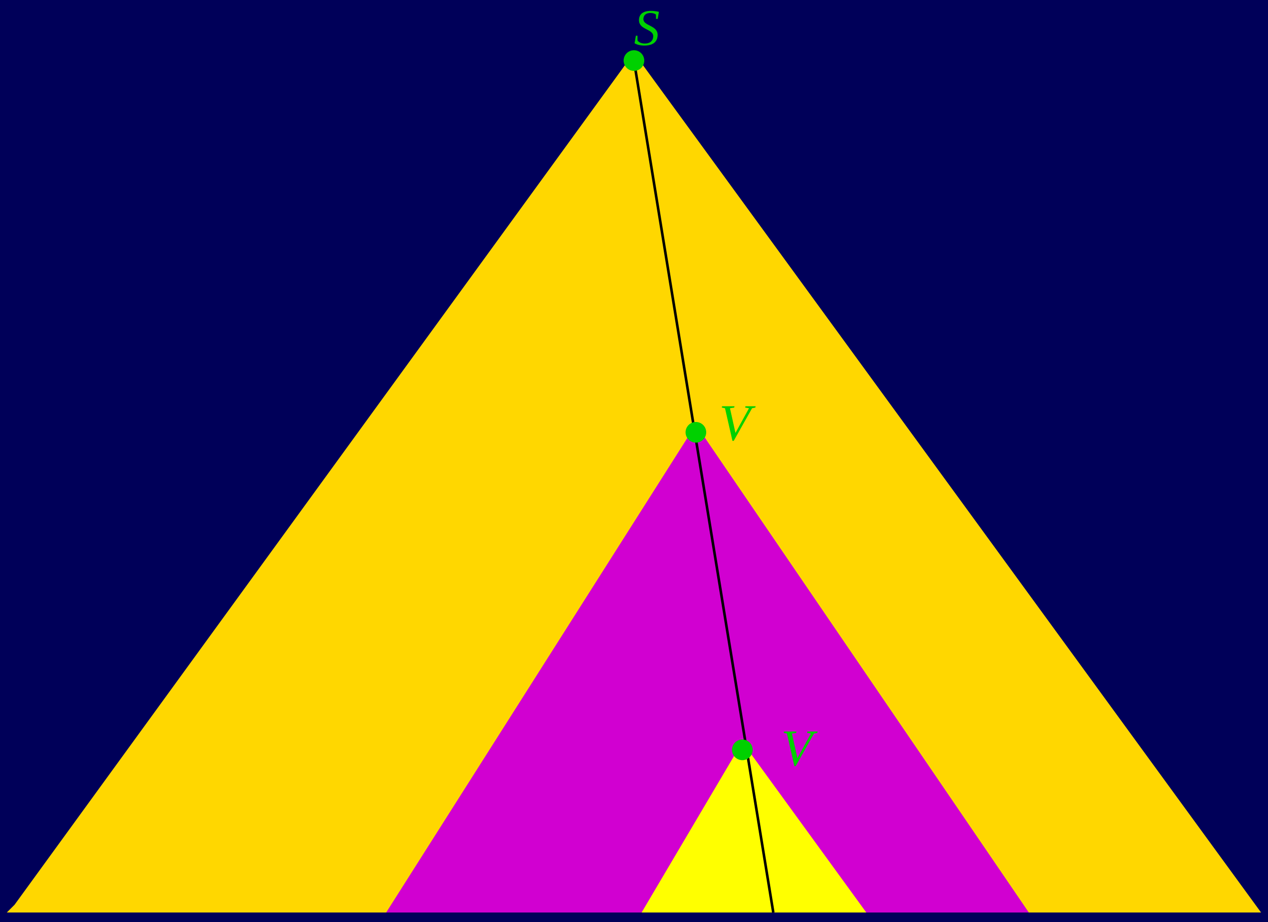
Any parse-tree deeper than the number of variables in the CFG must have some variable occur at least twice on its deepest path:



This provides “pluggable” subtrees!

Idea

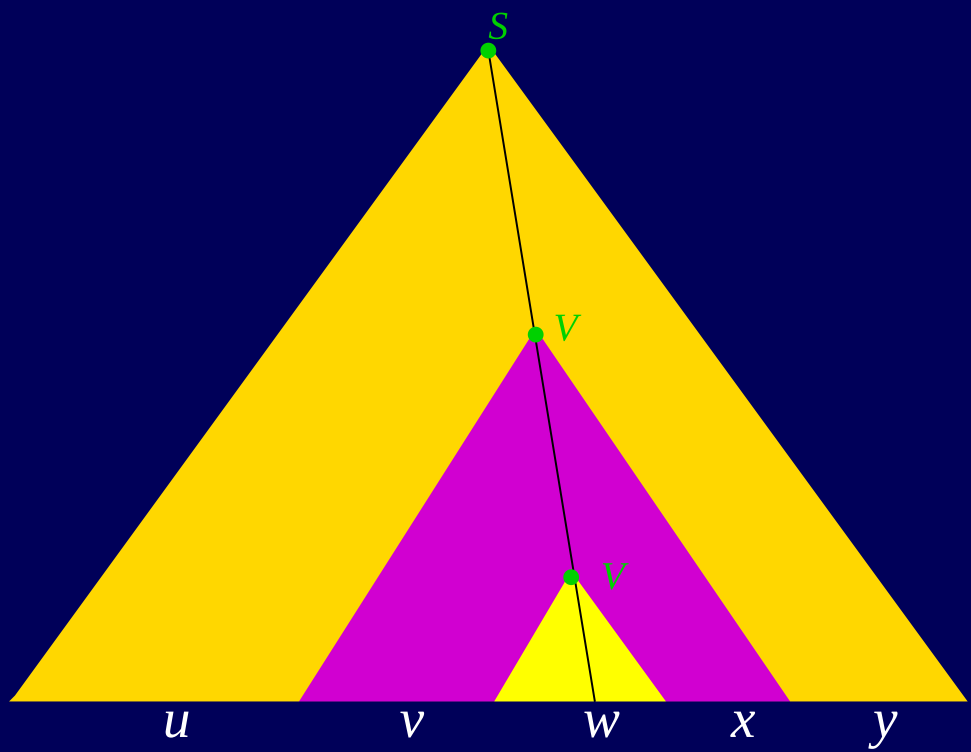
Any parse-tree deeper than the number of variables in the CFG must have some variable occur at least twice on its deepest path:



This provides “pluggable” subtrees!

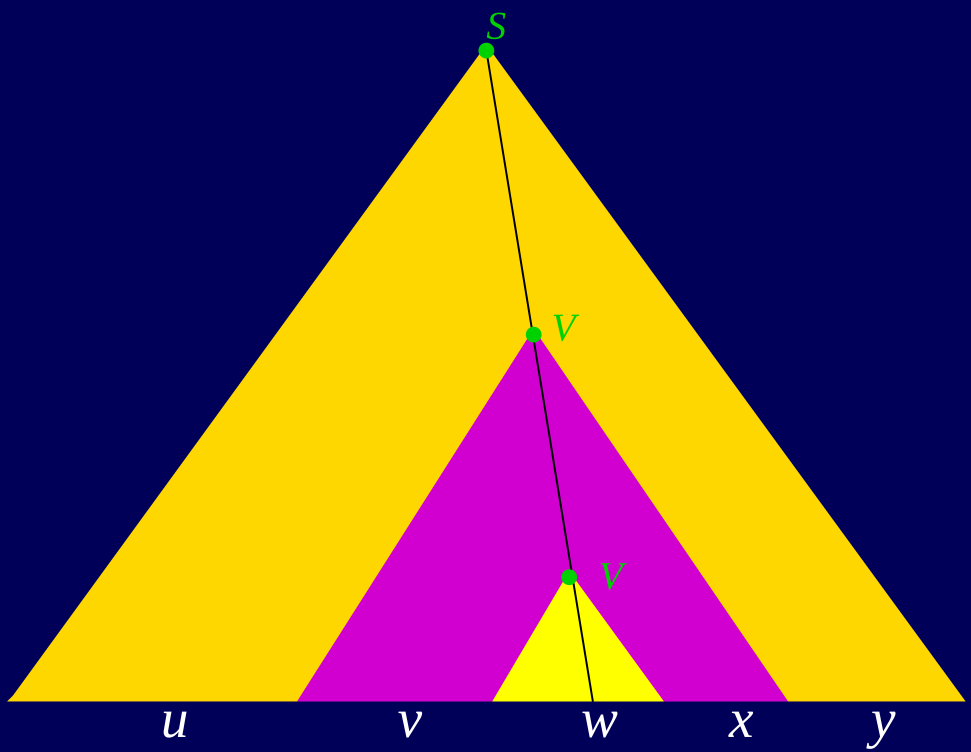
Pumping it

Parse tree

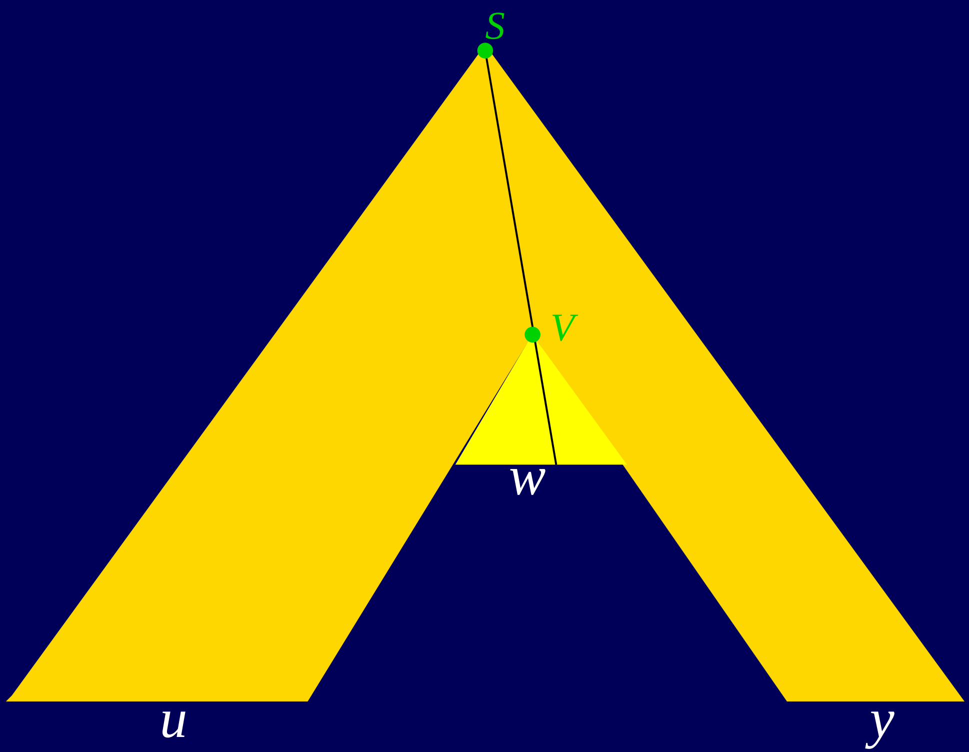


Pumping it

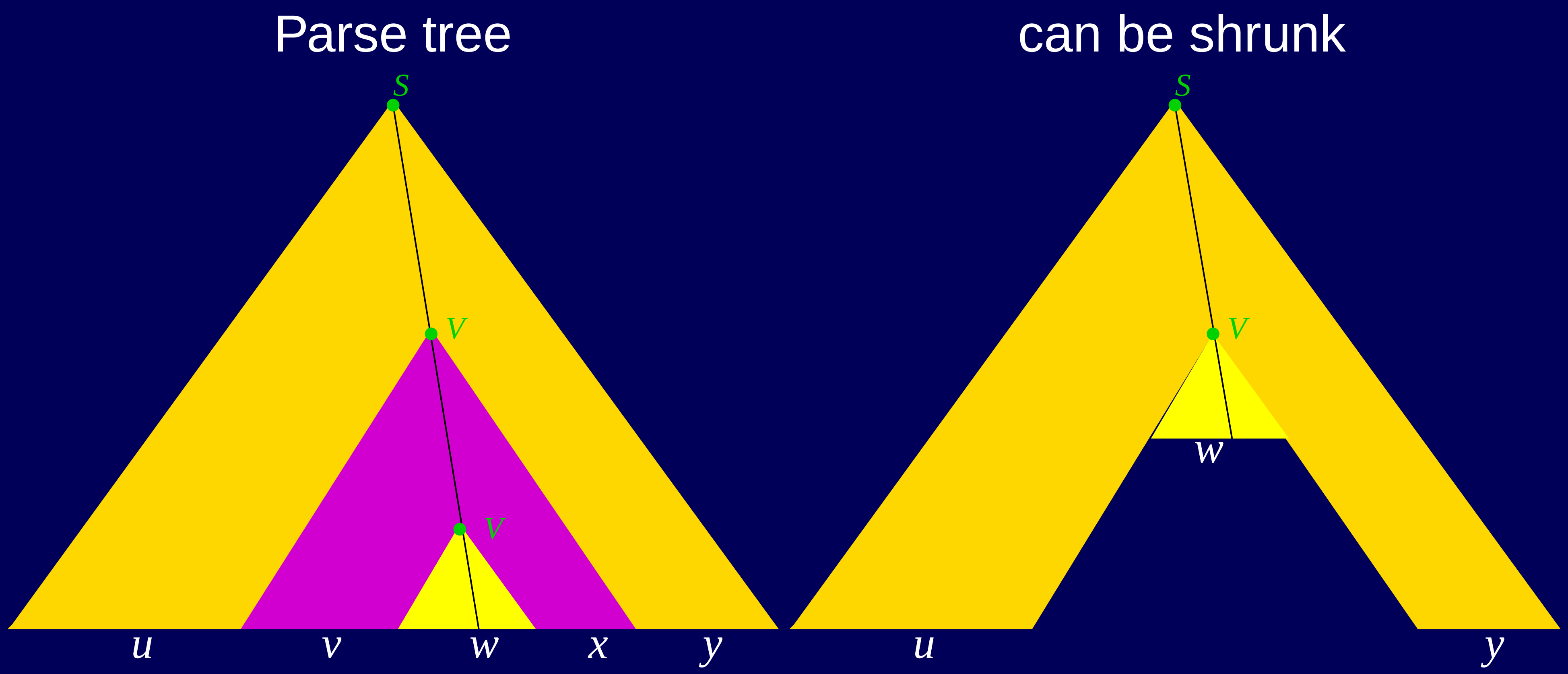
Parse tree



can be shrunk

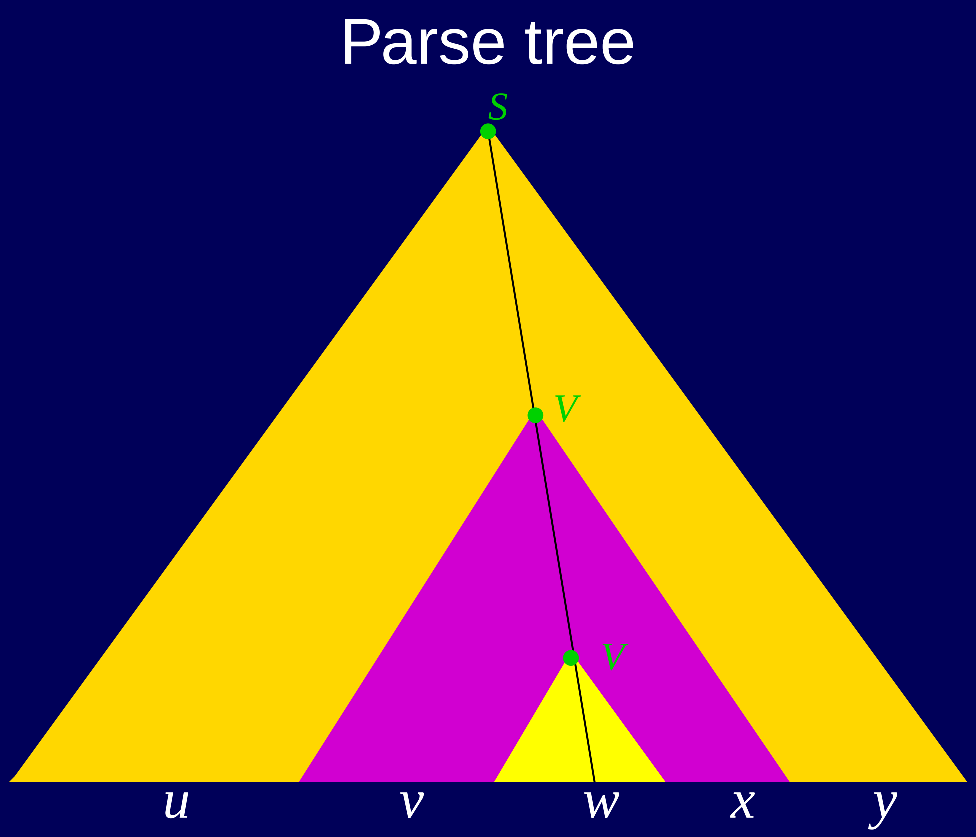


Pumping it

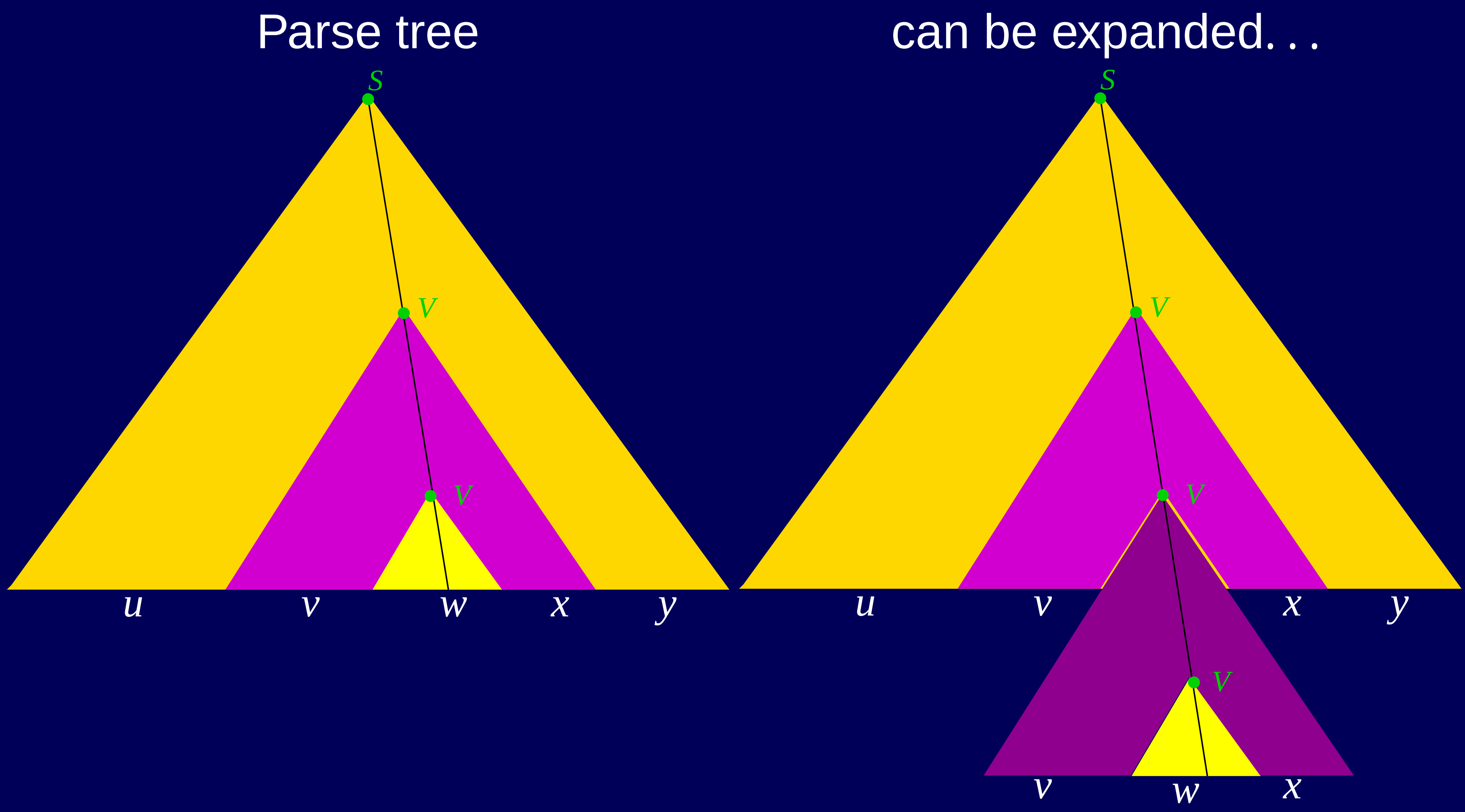


Caution: To really change yield of the parse tree, we have to make sure that $v \neq \varepsilon$ or $x \neq \varepsilon$.

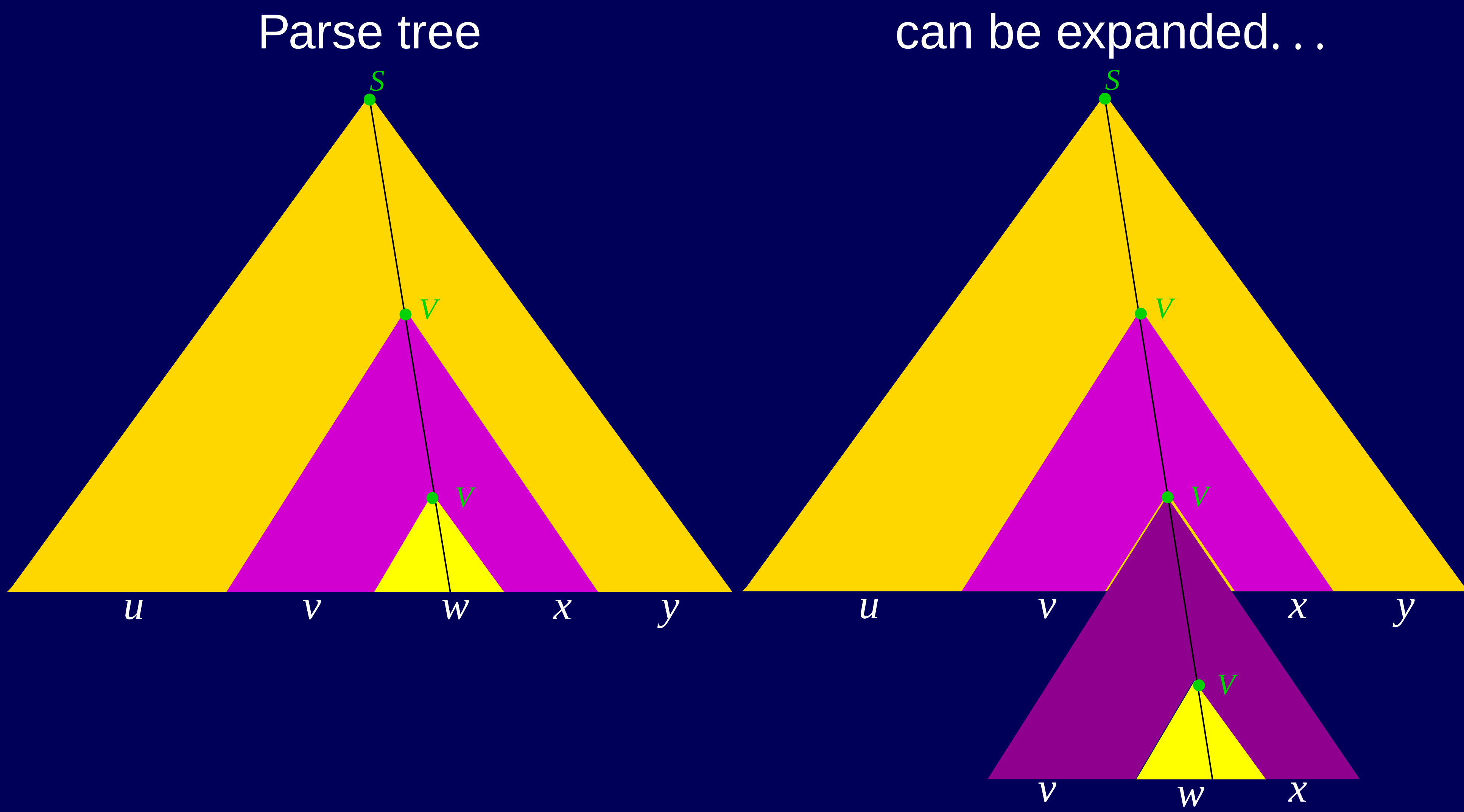
Pumping it



Pumping it



Pumping it



Again, change of yield depends on $vx \neq \varepsilon$.

Ensuring $v \neq \varepsilon$ or $x \neq \varepsilon$

can be done by using Chomsky-normal-form (CNF) grammars:

- they don't contain nullable symbols,
- the production used for the upper V in the parse tree must be of the form

$$V \to V_1 V_2$$

  as it did not directly derive a terminal

$\Rightarrow$ the lower V must be derived from either V_1 or V_2

- in the first case, $x \neq \varepsilon$ as V_2 is non-nullable,
- in the second case, $v \neq \varepsilon$ as V_1 is non-nullable.

The pumping lemma for CFLs

Thm: For any CFL L there is constant $n \in \mathbf{N}$ such that any string $z \in L$ with $|z| \geq n$ can be split into $z = uvwxy$ with

1. $|vwx| \leq n$,
2. $vx \neq \varepsilon$,
3. $\forall i \in \mathbf{N} \bullet uv^iwx^iy \in L$,

The pumping lemma for CFLs

Thm: For any CFL L there is constant $n \in \mathbb{N}$ such that any string $z \in L$ with $|z| \geq n$ can be split into $z = uvwxy$ with

1. $|vwx| \leq n$,
   i.e. we find the pumpable portion in a "short" substring,
2. $vx \neq \varepsilon$,
   i.e. at least one of v and x is non-trivial,
3. $\forall i \in \mathbb{N} \bullet uv^i wx^i y \in L$,
   i.e. v and x can be simultaneously "pumped".

An auxiliary lemma on CNF grammars

Lem: If $G = (V, T, P, S)$ is a CNF grammar and $w \in T^*$ is the yield of a parse tree of G of depth n then $|w| \leq 2^{n-1}$.

An auxiliary lemma on CNF grammars

Lem: If $G = (V, T, P, S)$ is a CNF grammar and $w \in T^*$ is the yield of a parse tree of G of depth n then $|w| \leq 2^{n-1}$.

Prf: By *complete* induction on n:

An auxiliary lemma on CNF grammars

Lem: If $G = (V, T, P, S)$ is a CNF grammar and $w \in T^*$ is the yield of a parse tree of G of depth n then $|w| \leq 2^{n-1}$.

Prf: By *complete* induction on n:

Induction hypothesis: For all $n < k$ it is true that the yield w of any parse tree of depth n satisfies $|w| \leq 2^{n-1}$, if w is terminal.

An auxiliary lemma on CNF grammars

Lem: If $G = (V, T, P, S)$ is a CNF grammar and $w \in T^*$ is the yield of a parse tree of G of depth n then $|w| \leq 2^{n-1}$.

Prf: By *complete* induction on n:

Induction hypothesis: For all $n < k$ it is true that the yield w of any parse tree of depth n satisfies $|w| \leq 2^{n-1}$, if w is terminal.

Induction step: Parse tree depth $n = k$. We deduce from the induction hypothesis that $|w| \leq 2^{k-1}$ for all terminal yields of parse trees of depth k.

An auxiliary lemma on CNF grammars

Lem: If $G = (V, T, P, S)$ is a CNF grammar and $w \in T^*$ is the yield of a parse tree of G of depth n then $|w| \leq 2^{n-1}$.

Prf: By *complete* induction on n:

Induction hypothesis: For all $n < k$ it is true that the yield w of any parse tree of depth n satisfies $|w| \leq 2^{n-1}$, if w is terminal.

Induction step: Parse tree depth $n = k$. We deduce from the induction hypothesis that $|w| \leq 2^{k-1}$ for all terminal yields of parse trees of depth k.

- **Case 1: $k = 0$.** There is no parse tree of depth 0 having a terminal yield. Thus, the conjecture is trivially true.

An auxiliary lemma on CNF grammars

Lem: If $G = (V, T, P, S)$ is a CNF grammar and $w \in T^*$ is the yield of a parse tree of G of depth n then $|w| \leq 2^{n-1}$.

Prf: By *complete* induction on n:

Induction hypothesis: For all $n < k$ it is true that the yield w of any parse tree of depth n satisfies $|w| \leq 2^{n-1}$, if w is terminal.

Induction step: Parse tree depth $n = k$. We deduce from the induction hypothesis that $|w| \leq 2^{k-1}$ for all terminal yields of parse trees of depth k.

- **Case 1: $k = 0$.** There is no parse tree of depth 0 having a terminal yield. Thus, the conjecture is trivially true.
- **Case 2: $k = 1$.** As G is of CNF and w is terminal, w can only be derived by a production $S \to a$ with $a \in T$. Hence, $|w| = 1 = 2^0 = 2^{k-1}$.

An auxiliary lemma on CNF grammars

Lem: If $G = (V, T, P, S)$ is a CNF grammar and $w \in T^*$ is the yield of a parse tree of G of depth n then $|w| \leq 2^{n-1}$.

Prf: By *complete* induction on n:

Induction hypothesis: For all $n < k$ it is true that the yield w of any parse tree of depth n satisfies $|w| \leq 2^{n-1}$, if w is terminal.

Induction step: Parse tree depth $n = k$. We deduce from the induction hypothesis that $|w| \leq 2^{k-1}$ for all terminal yields of parse trees of depth k.

- **Case 1:** $k = 0$. There is no parse tree of depth 0 having a terminal yield. Thus, the conjecture is trivially true.
- **Case 2:** $k = 1$. As G is of CNF and w is terminal, w can only be derived by a production $S \to a$ with $a \in T$. Hence, $|w| = 1 = 2^0 = 2^{k-1}$.
- **Case 3:** $k > 1$. As tree depth $k > 1$, the root of the parse tree uses a production, which is of the form $S \to V_1V_2$ because G is of CNF. As the overall tree depth is k, the subtrees rooted at V_1 and V_2 have depth $< k$, s.t. the induction hypothesis applies to them.
  Thus, $|w_1| \leq 2^{k-2} \geq |w_2|$ holds for the yields w_i of these subtrees. Now, $w = w_1w_2$ such that $|w| = |w_1| + |w_2| \leq 2^{k-2} + 2^{k-2} = 2^{k-1}$.

Pumping lemma as a proof scheme

Given a language L deemed to be non-CFL, proceed as follows:

1. Take arbitrary $n \in \mathbb{N}$,
2. provide a construction of z depending on n,
3. arbitrarily break z into $uvwxy$ subject to the constraints
   (a) $|vwx| \leq n$,
   (b) $vx \neq \varepsilon$,
4. pick $i \in \mathbb{N}$ depending on u, v, w, x, y and n such that $uv^iwx^iy \notin L$.

This constitutes a proof of L being not context-free.

Pumping lemma as a proof scheme

Given a language L deemed to be non-CFL, proceed as follows:

1. Take arbitrary $n \in \mathbf{N}$,
   (Selection of n is *not* under your control — you have to accept any n.)
2. provide a construction of z depending on n,
   (*You* select z.)
3. arbitrarily break z into $uvwxy$ subject to the constraints
   (a) $|vwx| \leq n$,
   (b) $vx \neq \varepsilon$,
   (Selection of u, v, w, x, y is *not* under your control — you have to accept any split that satisfies the two constraints.)
4. pick $i \in \mathbf{N}$ depending on u, v, w, x, y and n such that $uv^iwx^iy \notin L$.
   (*You* select i.)

This constitutes a proof of L being not context-free.

Closure properties of CFLs

Substitutions

Idea: *Generalize* the notion of *homomorphism* by substituting a full CFL (instead of just a word) for each terminal symbol.

Def: Given a set T of terminals, a substitution for T is a mapping $s : T \longrightarrow \mathit{CFL}$.

Given $w \in T^*$ and a substitution s on T,

$$s(w) \stackrel{\text{def}}{=} \{v_1 v_2 \ldots v_{|w|} \mid v_i \in s(w_i) \text{ for all } i \leq |w|\} \ ,$$

i.e. $s(w)$ is the concatenation of the languages $s(w_1), s(w_2), \ldots$.

Given $L \subseteq T^*$ and a substitution s on T,

$$s(L) \stackrel{\text{def}}{=} \bigcup_{w \in L} s(w) \ .$$

Closure under substitution

Thm: If L is a CFL over alphabet T and s is a substitution for T then $s(L)$ is a CFL.

Prf: Essentially, we take a CFG G for L and replace each terminal a by the start symbols of a CFG for $s(a)$.

Closure under substitution

Thm: If L is a CFL over alphabet T and s is a substitution for T then $s(L)$ is a CFL.

Prf: Essentially, we take a CFG G for L and replace each terminal a by the start symbols of a CFG for $s(a)$.

Let $G = (V, T, P, S)$ and $G_a = (V_a, T_a, P_a, S_a)$ be a CFG for $s(a)$ for each $a \in T$. Then $G' = (V', T', P', S)$ generates $s(L)$, where

- V' is the *disjoint* union of V and all V_a's,
- $T' = \bigcup_{a \in T} T_a$,
- P' consists of
  1. the productions from P, but with each terminal $a \in T$ being replaced by S_a (actually a homomorphism on the productions),
  2. $\bigcup_{a \in T} P_a$.

Implications of substitution closure

Cor: The CFLs are closed under

- union,
- concatenation,
- Kleene closure (*) and positive closure ($^+$),
- homomorphism.

Implications of substitution closure

Cor: The CFLs are closed under

- union,
- concatenation,
- Kleene closure (*) and positive closure ($^+$),
- homomorphism.

Prf: For the first three items, we state regular (and thus context-free) languages plus a substitution that maps them into the desired language:

1. $a + b$,
2. ab,
3. a^* and aa^*, respectively.

Applying the substitution $s(a) = L$ and $s(b) = M$ to these RLs, we obtain

1. $L \cup M$,
2. LM,
3. L^* and L^+.

Implications of substitution closure

Cor: The CFLs are closed under

- union,
- concatenation,
- Kleene closure (*) and positive closure ($^+$),
- homomorphism.

Prf: For the first three items, we state regular (and thus context-free) languages plus a substitution that maps them into the desired language:

1. $a + b$,
2. ab,
3. a^* and aa^*, respectively.

Applying the substitution $s(a) = L$ and $s(b) = M$ to these RLs, we obtain

1. $L \cup M$,
2. LM,
3. L^* and L^+.

Homomorphism, finally, is a special case of substitution.

Hence, substitution closure implies all these closure properties.

Non-closure under intersection

Thm: The CFLs are *not* closed under intersection.

N.B.: This is in contrast to the RLs, which are closed under intersection.

Prf: By contraposition:

$M = \{a^n b^n c^m \mid n, m \in \mathbb{N}\}$ and $N = \{a^n b^m c^m \mid n, m \in \mathbb{N}\}$ are CFLs. If the CFLs were closed under intersection then $\{a^n b^n c^n \mid n \in \mathbb{N}\} = M \cap N$ were a CFL, which it is not.

Implications of non-closure under intersection

Cor: The CFLs are not closed under

- complement,
- difference.

Prf: By contraposition:

If the CFLs were closed under complement, then they were also closed under intersection, as $M \cap N = \overline{\overline{M} \cup \overline{N}}$.

If the CFLs were closed under difference, then they were also closed under complement, as T^* is a CFL and $\overline{L} = T^* \setminus L$.

Closure under reversal

Thm: The CFLs are closed under reversal of words.

Prf: Take a CFG for the language and reverse the bodies of all productions.

Closure under intersection with RL

Thm: The CFLs are closed under intersection with a regular language.
I.e., $L \cap R$ is a CFL if L is a CFL and R is an RL.

Prf: Let $P = (Q_P, \Sigma, \Gamma, \delta_P, q_P, Z_0, F_P)$ be a PDA accepting L by final state and let $A = (Q_a, \Sigma, \delta_A, q_A, F_A)$ be a DFA for R.
The PDA $Q' = (Q_P \times Q_a, \Sigma, \Gamma, \delta, (q_P, q_A), Z_0, F_P \times F_a)$ with

$$\delta((q,r),a,X) \stackrel{\text{def}}{=} \left\{ ((q',r'),\gamma) \;\middle|\; \begin{array}{cl} & (q',\gamma) \in \delta_P(q,a,X) \\ \wedge & r' = \hat{\delta}_Q(r,a) \end{array} \right\}$$

then accepts $L \cap R$ by final state.

Closure under intersection with RL

Thm: The CFLs are closed under intersection with a regular language.
I.e., $L \cap R$ is a CFL if L is a CFL and R is an RL.

Prf: Let $P = (Q_P, \Sigma, \Gamma, \delta_P, q_P, Z_0, F_P)$ be a PDA accepting L by final state and let $A = (Q_a, \Sigma, \delta_A, q_A, F_A)$ be a DFA for R.
The PDA $Q' = (Q_P \times Q_a, \Sigma, \Gamma, \delta, (q_P, q_A), Z_0, F_P \times F_a)$ with

$$\delta((q,r),a,X) \stackrel{\text{def}}{=} \left\{ ((q',r'),\gamma) \;\middle|\; \begin{array}{rr} & (q',\gamma) \in \delta_P(q,a,X) \\ \wedge & r' = \hat{\delta}_Q(r,a) \end{array} \right\}$$

then accepts $L \cap R$ by final state.

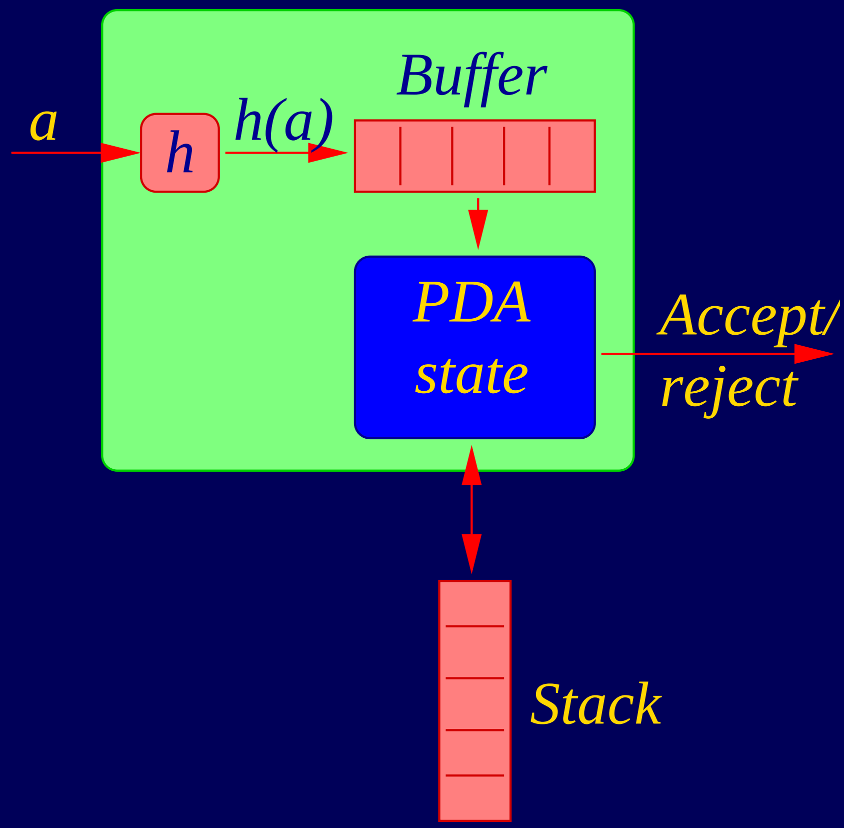
Cor: If L is a CFL and R is an RL, then $L \setminus R$ is a CFL.

Prf: $L \setminus R = L \cap \overline{R}$. Thus, $L \setminus R$ is a CFL because of closure of the CFLs under intersection with RLs and because of closure of the RLs under complement.

Closure under inverse homomorphism

Thm: The CFLs are closed under reverse homomorphism.

Prf: We add a length $max\{|h(a)| \mid a \in T\}$ buffer to a PDA for the CFL:



- Buffer is part of finite state set,
- whenever it is empty, it can be filled with $h(\text{next input})$, thereby keeping the stack and the original PDA's state,
- when buffer is not empty, the original PDA's non-ε moves can be performed on the frontmost element of the buffer, thereby removing that element from the buffer,
- the original PDA's ε moves can always be performed, leaving the buffer intact.