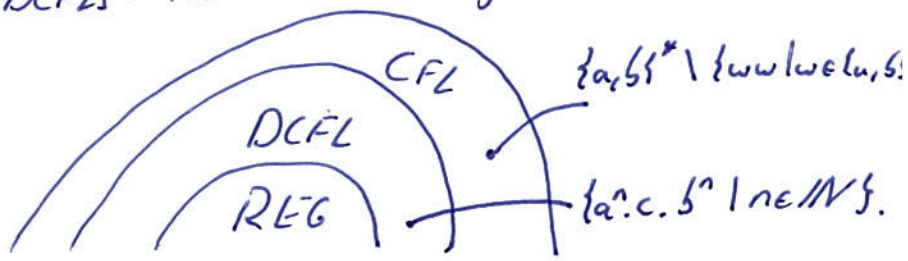
12. Deterministic Context-free Languages

Goal:
- Introduce deterministic context-free languages (DCFLs) that are important in compiler construction.
- Study their closure properties, in particular establish closure under complementation.

Note:
- By closure under complement, DCFLs cannot be as expressive as full CFLs.
- Every regular language is a DCFL by Rabin & Scott. Moreover, there are DCFLs that are not regular.

Together, we have



- In this section, we use PDAs of the form

$$M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, Q_F)$$

with $\delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times \Gamma \times \Gamma^* \times Q$,

so we definitely read the stack in every move and therefore need an initial stack symbol $Z_0 \in \Gamma$.

- DCFLs are defined via deterministic pushdown automata (DPDAs).
The idea of determinism is that the labelled transition relation $(q_1, \gamma_1) \xrightarrow{w} (q_2, \gamma_2)$ among configurations is a function, like for DFAs.

Definition:
A PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, Q_F)$ is deterministic, if for all $q \in Q$, $a \in \Sigma$, and $A \in \Gamma$, δ contains at most one transition of the form $q \xrightarrow[A/\gamma']{a} q'$ or $q \xrightarrow[A/\gamma'']{\varepsilon} q''$.

To explain the condition:

- If there is a labelled transition with Π at the top-of-stack, then there is no ε-transition reaching to Π at the top-of-stack.

  Note that by contraposition this implication also says:

  If there is an ε-transition, then there are no labelled transitions.

- There is at most one ε-transition reaching to Π at the top-of-stack.
- There is at most one a-labelled transition reaching to Π.

12.1 Closure under Complement

Idea: Like for finite automata, swap final and non-final states.

Problems: (1) The DPDT may not read the whole input string w.

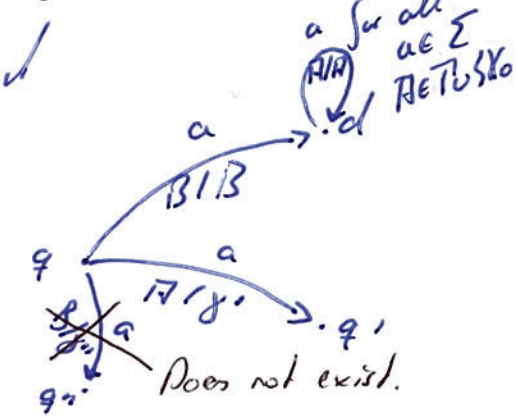
- Why? ↳ It gets stuck in a configuration where no moves are possible or
  ↳ it makes an infinity of ε-moves without using further input symbols.
- As a consequence, the DPDT does not accept any word that has w as a prefix.
  Thus, the complement DPDT should accept every word with w as a prefix.
- But if we simply swap final and non-final states, the resulting DPDT would still not move beyond w.

(2) After having seen the input word, the DPDT may make ε-moves, some leading to final states, others leading to non-final states.
In this case, swapping the states still lets us accept the input, and this is wrong.

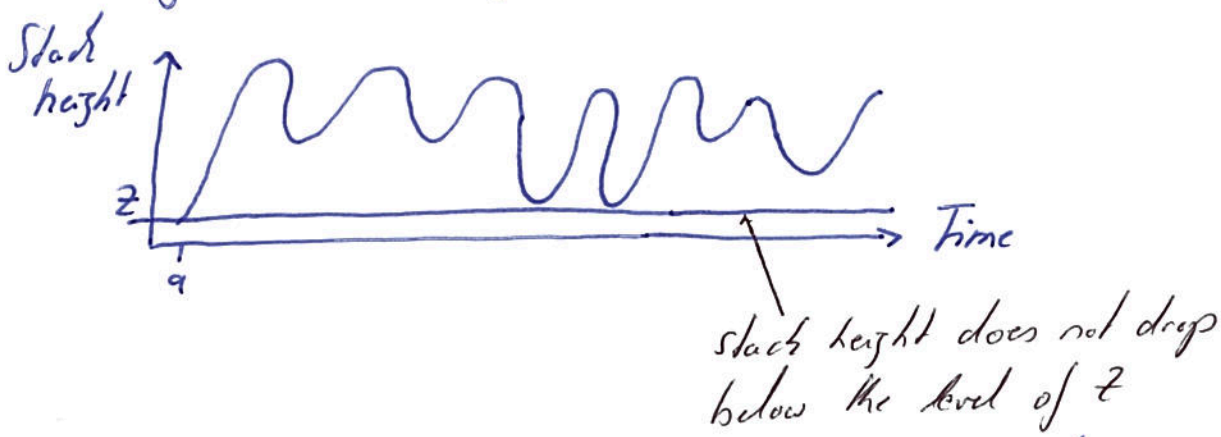
- To address the first problem, we show that, given a DPDA M, we can construct a DPDA M' that will never reach a configuration from which it will not read another input symbol.
- The construction of M' given in the following lemma still contains a condition for which decidability is unclear. The construction is therefore not yet known to be computable.
- We address the required decidability in Section 12.2.

Lemma:
Let M be a DPDA. There is (and with Section 12.2 we can also compute it) a DPDA M' with $L(M) = L(M')$ and where M' is guaranteed to read the entire input.

Idea: - We assume that for every reachable configuration (q, x) and input symbol $a \in \Sigma$, M has a next move. Otherwise, we add an endmarker $\$$ on the stack to prevent M from emptying the stack and getting stuck without having read the entire input.
In addition, we add a dead state d so that for any combination of state, stack symbol, and input symbol for which there is no move, we go to d (with that input). From d we move to d by reading any input and without changing the stack.
State d is not final.



- Now M has a next move in every configuration.
The only way M could not read the entire input
is by an infinite sequence of ε-moves from some configuration.
If in state q with Z at the top-of-stack
M makes an infinite sequence of ε moves
<u>without erasing the level of Z</u>:



↳ Let M go to d if none of the states was found.

↳ Let M go to a fresh final state f,
if M entered some accepting state during the ε-sequence.
From f we go to d with another ε-transition.
The fresh state ensures we accept the current prefix
(as we should by the final state).

<u>Note</u>: The side condition <u>without erasing the level of Z</u> is important.
It makes sure the condition is independent
of the prefix of the computation, in particular
independent of what is currently on the stack.
This is needed to ensure well-definedness.

<u>Construction</u>: Let $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, Q_F)$.

We define $\mathcal{A}' := (Q \cup \{q_0'\}, d, \{f\}, \Sigma, \Gamma \cup \{X_0\}, q_0', X_0, \delta', Q_F \cup \{f\})$.

The transitions are the following:

(1) Put Z_0 on top of X_0:

$$q_0' \xrightarrow[X_0/X_0 Z_0]{\varepsilon} q_0.$$

(2) Enter the dead state if no move is possible:

- For all $q \in Q$, $a \in \Sigma$, $A \in \Gamma$:
  If there are no $\gamma \in \Gamma^*$, $q' \in Q$ with
  $$q \xrightarrow[A/\gamma]{a} q' \quad \text{or} \quad q \xrightarrow[A/\gamma]{\varepsilon} q'$$
  then
  $$q \xrightarrow[A/A]{a} d \in \delta'.$$

- Moreover, we actively test whether the stack is empty (and therefore no move is possible):
  For all $q \in Q$: $q \xrightarrow[X_0/X_0]{\varepsilon} d \in \delta'$.

(3) Loop in d and read the input:

$$d \xrightarrow[Z/Z]{a} d \in \delta' \text{ for all } a \in \Sigma,\ Z \in \Gamma.$$

(4) For all $q \in Q$, $Z \in \Gamma$:
If for all $i \in \mathbb{N}$ there are $q_i \in Q$, $\gamma_i \in \Gamma^*$ with $(q, Z) \xrightarrow{\varepsilon}{}^{i} (q_i, \gamma_i)$, (*)
then $q \xrightarrow[Z/Z]{\varepsilon} d \in \delta'$, provided none of the q_i is final
$q \xrightarrow[Z/Z]{\varepsilon} f \in \delta'$, if at least one q_i is final.

(*) Note that starting from (q, Z) captures the idea that the stack level never drops below Z.

Note: We have not yet claimed that this construction is computable.
How to check the "for all $i \in \mathbb{N}$" condition algorithmically,
we will study in Section 12.2.

(5) $f \xrightarrow[Z/\Gamma]{\varepsilon} d \in \delta'$ for all $Z \in \Gamma \cup \{X_0\}$.

(6) For all $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, and $Z \in \Gamma$
where δ' has not been defined:

$$q \xrightarrow[Z/\gamma]{a} q' \in \delta' \text{ iff } q \xrightarrow[Z/\gamma]{a} q' \in \delta$$

for all $\gamma' \in \Gamma^*$ and $q' \in Q$.

Correctness:

- $L(M') = L(M)$ is not difficult to show.
- We argue that M' is guaranteed to read the entire input.
Assume this was not the case and, upon input $w = w_1 \cdot w_2 \in \Sigma^*$,
we have
$$(q_0', X_0) \xrightarrow{w_1} (q, X_0 Z_1 \dots Z_n)$$
so that from $(q, X_0 Z_1 \dots Z_n)$ no more symbols of w are consumed.
- By Rule (2), it cannot be the case that M' gets stuck.
- Therefore, M' must make an infinite sequence of ε-moves.
- If in this sequence M' never erased the stack level of Z_n,
Rule (4) would have sent M' to stack d,
in which it would have read the entire input. $\lightning$
Therefore M' eventually pops the level of Z_n from the stack.

- We repeat the argument for Z_{n-1} to Z_1.
  Eventually, this leads us to a configuration (q', X_0).
  By Rule (2), M' is again sent to d
  and reads the entire input. ↯ There cannot be
  an infinite sequence
  of ε-moves. □

It remains to deal with the second problem:
After having read the entire input, the DPDT may make
a sequence of ε-moves, entering both final and non-final states.

Solution: • Add a second component to the state
recording whether a final state of the original DPDT has been entered
since the last time a non-ε-input was read.
- If no such final state has been entered,
  the complement DPDT enters a final state
  just before it is ready to read the next input symbol.
- The just before condition is important,
  otherwise we are not sure to avoid final states in the ε-future.

Theorem:
If L is a DCFL, so is $\overline{L}$.

Proof: Let $L = L(M)$ with $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, Q_F)$ a DPDT
that is guaranteed to read the entire input.
We define
$$M' := (Q \times \{1,2,3\}, \Sigma, \Gamma, q_0', Z_0, \delta', Q \times \{3\}).$$

Here, $q_0' := \begin{cases} (q_0, 1), & \text{if } q_0 \in Q_F \\ (q_0, 2), & \text{if } q_0 \notin Q_F \end{cases}$

The purpose of h in (q, h) is to record, between non-ε-inputs, whether or not A has entered a final state:

$\underline{h=1}$: A has entered a final state since the last non-ε-input.

$\underline{h=2}$: A has not entered a final state since the last non-ε-input.

- If $h=1$ when A' reads a non-ε-input, it simulates the move of A and changes h to 1 or 2, depending on whether the target state of the transition is final.
- If $h=2$ when A' reads a non-ε-input, it first changes h to 3 via ε in order to accept, and then simulates the move of A, changing h to 1 or 2.

Formally:

(1) If $h=1$ or $h=2$ and $q \xrightarrow[Z/\gamma]{\varepsilon} p \in \delta$,

then $(q, h) \xrightarrow[Z/\gamma]{\varepsilon} (p, h') \in \delta'$

where $h' := 1$ if $h=1$ or $p \in Q_F$, otherwise $h' := 2$.

(2) If $q \xrightarrow[Z/\gamma]{a} p \in \delta$, then $(q, 2) \xrightarrow[Z/Z]{\varepsilon} (q, 3) \in \delta'$.

Moreover, $(q, 3) \xrightarrow[Z/\gamma]{a} (p, h)$ and $(q, 1) \xrightarrow[Z/\gamma]{a} (p, h)$.

In both cases, $h=1$ or $h=2$ for $p \in Q_F$ and $p \notin Q_F$, respectively. □

Corollary: Every DCFL is accepted by a DPDA that, in a final state, makes no move on ε.

Proof: The $(q,3)$ states in the construction above have no ε-moves. □

Example (Application of the result):

- $L = \{a,b\}^* \setminus \{ww \mid w \in \{a,b\}^*\}$
  is not a DCFL.
  If it was, so was $\bar{L} = \{ww \mid w \in \{a,b\}^*\}$.
  We know that $\bar{L}$ is not even context-free. □
  Interestingly, L is a CFL.

- $L = \{a^i b^j c^k \mid i=j \text{ or } j=k\}$
  is not a DCFL.
  If the language was a DCFL, then so was $\bar{L}$.
  But then
  $$\bar{L} \cap a^*b^*c^* = \{a^i b^j c^k \mid i \neq j \text{ and } j \neq k\}$$
  is a CFL. { By Ogden's lemma, this is not a CFL. □

12.2 Computability

Note: · The above result shows that the deterministic context-free languages are closed under complement.
· It does not show that we can compute an automaton for the complement.

This may happen:

It is known that the downward closure $L\downarrow$ (wrt. scattered subwords $a_1 \dots a_n \leq u_0 a_1 u_1 \dots u_{n-1} a_n u_n$) of every language $L \subseteq \Sigma^*$ is regular.

This means one cannot compute an automaton for $L\downarrow$,

for otherwise one could solve the halting problem by checking $L(A) \neq \emptyset$.

Goal: Show that Property (4) is decidable
and therefore the above construction is computable.

Approach: Normalize the given DPDA so that
it never changes the top-of-stack symbol
(while staying at that stack level).

Lemma: For every DPDA M, one can construct a DPDA M'
with $L(M) = L(M')$ and where $q \xrightarrow[A/\gamma]{a} q'$ implies

- γ is ε (pop)
- γ is A (stay at stack level and do not change the top-of-stack)
- γ is AB (push B, keep A at the former level).

Idea: Keep the current top-of-stack in the finite control,
thus using states $Q \times \Gamma$.

Proof: Let $L = L(M)$ with $M = (Q, \Sigma, \Gamma, q_0, X_0, \delta, Q_F)$.

We define

$$M' := (Q \times \Gamma, \Sigma, \Gamma, (q_0, X_0), X_0, \delta', Q_F \times \Gamma).$$

We already assume right-hand sides have length at most 2 in δ.

The transitions are as follows:

(1) If A pops the stack, A' pops the stack, picking up the symbol for its control:

If $q \xrightarrow[X/\varepsilon]{a} p \in \delta$ with $a \in \Sigma \cup \{\varepsilon\}$,

then for all $Y \in \Gamma$: $(q, X) \xrightarrow[Y/\varepsilon]{a} (p, Y) \in \delta'$.

(2) If A changes the top-of-stack, A' records the change in its control state (but does not modify the top-of-stack):

If $q \xrightarrow[X/Y]{a} p \in \delta$ with $a \in \Sigma \cup \{\varepsilon\}$,

then for all $Z \in \Gamma$: $(q, X) \xrightarrow[Z/Z]{a} (p, Y) \in \delta'$.

(3) If the stack of A grows, A' pushes an appropriate symbol:

If $q \xrightarrow[X/YZ]{a} p \in \delta$ with $a \in \Sigma \cup \{\varepsilon\}$,

then for all $W \in \Gamma$: $(q, X) \xrightarrow[W/WY]{a} (p, Z) \in \delta$. □

Consequence of the construction: The top-of-stack symbol does not matter. Only the control state determines whether (only) ε-moves or (only) non-ε-moves are made. These moves are then enabled for all stack symbols.

To get decidability of Property (4), we compute the following information.

Lemma:

Consider $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, Q_F)$ a DPDA in normal form.

For all $q, p \in Q$, $Z \in \Gamma$ we can decide whether

(1) $(q, Z) \xrightarrow{\varepsilon}{}^{*} (p, \varepsilon)$

(2) $(q, Z) \xrightarrow{\varepsilon}{}^{*} (p, \gamma)$ for some $\gamma \in \Gamma^{+}$.

This indeed solves (4).
The difficult part is to check for an infinite run.
Checking whether an accepting state is reachable works with the lemma (use both items).

Proposition:

Consider $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, Q_F)$ a DPDA in normal form.

The following problem is decidable.

Given: $q \in Q$, $Z \in \Gamma$.

Question: For every $i \in \mathbb{N}$, are there $q_i \in Q$, $\gamma_i \in \Gamma^{*}$ with $(q, Z) \xrightarrow{\varepsilon}{}^{i} (q_i, \gamma_i)$.

Proof:

(1) We first check, for each state $p \in Q$ that only has non-ε-transitions (or no transition at all), whether $(q, Z) \xrightarrow{\varepsilon}{}^{*} (p, \gamma)$ for some $\gamma \in \Gamma^{*}$.
Each query is decidable by the above lemma (both items).
We do a finite number of such queries.

- Note that, by the normal form, a state either only has ε-transitions or only non-ε-transitions.
The top-of-stack does not matter.

- Moreover, since the automaton is deterministic, if the state is reachable via ε, the automaton will definitely go there.

Together, if the check is successful for one of the states p, we return <u>false</u>.

(2) If all the previous checks fail, we know that the automaton will either get stuck after some ε-transitions or run into an infinite ε-sequence from (q, Z).

<u>Claim:</u>

The automaton runs into an infinite ε-sequence from (q, Z) iff the stack is never emptied.

<u>Proof:</u>

„⇒" By the normal form, if the automaton runs into an infinite ε-sequence, Z remains at the bottom of the stack.

„⇐" If the automaton eventually halts, we reached a configuration with a cached state that

(a) either only has non-ε-transitions or no transition at all

(b) or it only (but definitely) has ε-transitions but none of them is enabled.

Because check (1) failed, we know (a) cannot be the case. Hence we are in Case (b).

By the normal form, if a control state has ε-transitions, it has these transitions for all stack symbols.
Hence, if all these transitions are disabled, the stack must be empty. □ Claim

We check, using the above Lemma, Item (1), whether the stack is ever emptied.
If so, we return *false*.
If not, we return *true*. □ Proposition

It remains to establish the lemma.

Proof (of the lemma above):

We construct relations

$$T_1, T_2 \subseteq Q \times \Gamma \times Q$$

so that for $i = 1, 2$

$(q, z, p) \in T_i$ iff Statement (i) in the Lemma is true for q, z, p.

Initially, $T_1 := \emptyset =: T_2$
We fill the relations using a least fixed point.

Base case: $(q, z, p) \in T_1$, if $q \xrightarrow[z/\varepsilon]{\varepsilon} p \in \delta$.

$(q, z, p) \in T_2$, if $q \xrightarrow[z/z]{\varepsilon} p \in \delta$

or $q \xrightarrow[z/z.\gamma]{\varepsilon} p \in \delta$.

Induction step: (1) Assume $q \xrightarrow{\varepsilon \,/\, Z \,/\, Y} r \in \delta$.

(1.a) If $(r, Y, s) \in T_1$ and $(s, Z, p) \in T_1$, then $(q, Z, p) \in T_1$.

(1.b) If $(r, Y, s) \in T_1$ and $(s, Z, p) \in T_2$, then $(q, Z, p) \in T_2$.

(1.c) If $(r, Y, p) \in T_2$, then $(q, Z, p) \in T_2$.

(2) Assume $q \xrightarrow{\varepsilon \,/\, Z \,/\, \varepsilon} r \in \delta$.

(2.a) If $(r, Z, p) \in T_1$, then $(q, Z, p) \in T_1$.

(2.b) If $(r, Z, p) \in T_2$, then $(q, Z, p) \in T_2$. □

With this fixed point, we are able to compute the complement DPDA.
This concludes the proof of our main result.

12.3 Further Closure Properties

Goal: Study the remaining operations.

Theorem:
The DCFLs are closed under inverse homomorphisms and under regular intersection.

Proof:
The automata-theoretic constructions for CFLs preserve determinism and therefore carry over to DCFLs. □

Theorem:
The DCFLs are not closed under homomorphisms, union, intersection, concatenation, or Kleene star.

Proof: Homework.