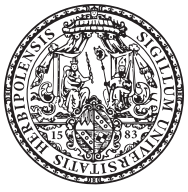
Formal Languages Defined by Recurrent Circuits

Diplomarbeit

von

Christian Reitwießner

29. August 2007
Betreuer: Prof. Dr. Klaus W. Wagner



Institut für Informatik
Julius-Maximilians-Universität Würzburg

Contents

1 Introduction

In theoretical computer science, a formal language is an arbitrary set of words, whereas a word is an arbitrary finite sequence of symbols from a fixed, finite set. Different operations over words and languages are considered in the theory of formal languages; the most prominent examples are the set-theoretic union ($\cup$) and intersection ($\cap$), the concatenation of words ($\cdot$) and the iteration (*, the arbitrarily repeated concatenation of a word to itself). The concatenation and thus also the iteration can also be extended to languages, one simply concatenates every word from one language with every word from another language. The class of the so-called *regular languages* is one of the very fundamental concepts in the theory of formal languages. It is the class of languages that can be constructed by starting with single symbols and applying just these mentioned operations. For example, the regular language represented by the regular expression $0 \cdot (0 \cup 1)^* \cap (0 \cup 1)^* \cdot 0$ contains all words over the alphabet $\{0, 1\}$ that start and end with 0.

In this work, we will extend these static constructions by recurrence; sets that have already been created can be re-used. We use circuits to model these recurrent expressions, but unlike normal logical circuits, the information on a wire of such a circuit is not zero or one but a set of words, and the gates do not compute boolean functions but apply one of the before mentioned operations to their inputs. If such a circuit is combinatoric (i.e. if its graph does not contain loops) then the language described by the circuit is regular, since these circuits can easily be transformed into a regular expression. So we will allow the circuits to be non-combinatoric and thus have to use some clocked manner to define the sets that are generated in the gates: Each gate starts with a very simple set of words and in each step it applies the operation to incoming languages and appends the result in its set of generated words.

After having defined these circuits formally, we will see that we can of course still construct all regular sets, but also all context-free languages and even a bit more than that. By limiting the set of possible operations, different language classes are obtained. The class $\mathrm{RC}(\cup, \cdot)$, for instance, is the class of languages generated by recurrent circuits that are allowed to use the operations of union and concatenation. For some of these classes, the equality to already known language-theoretic classes can be shown. That way, we for example gain an alternative characterization for the class of context-free languages: It is equal to the just mentioned class $\mathrm{RC}(\cup, \cdot)$. It also turns out that $\mathrm{RC}(\cup, \cap, \cdot)$, the unlimited case, is equal to the set of languages generated by so-called con-

junctive grammars, defined only recently by Okhotin [Okh01]. For two other interesting classes, we were not able to find known equal classes. These are $\mathrm{RC}(\cdot)$ and $\mathrm{RC}(\cap, \cdot)$ and they are classes of languages generated by restrictions of context-free and conjunctive grammars, respectively. We will show that the class $\mathrm{RC}(\cdot)$ can be defined using a certain kind of pushdown-automaton. We were not able to separate the class $\mathrm{RC}(\cap, \cdot)$ from the unlimited class. Although they share many properties, it would be quite surprising if equality could be shown since union is a powerful operation.

At the end of this thesis, we will present and discuss the obtained results. As we especially focus on closure properties and inclusions for the defined classes, we will list the closure properties of most of the classes (Table 1) and give an inclusion diagram (Figure 8) of all the classes and also their relation to important language- and complexity-theoretic classes.

2 Preliminaries

2.1 Sets, Words, Operations

We start with some basic definitions. Let $\mathbb{N} \stackrel{\text{df}}{=} \{0, 1, 2, \dots\}$ be the *natural numbers*. For any set A we denote the *power set* of A as $\mathcal{P}(A) \stackrel{\text{df}}{=} \{B \mid B \subseteq A\}$. Two sets A, B are called *disjoint* if $A \cap B = \emptyset$. The set-theoretic subtraction is denoted by $A - B \stackrel{\text{df}}{=} \{x \in A \mid x \notin B\}$. For $n \in \mathbb{N}$ and sets $A_1, A_2, \dots, A_n$ we define $A_1 \times A_2 \times \cdots \times A_n \stackrel{\text{df}}{=} \{(a_1, a_2, \dots, a_n) \mid a_i \in A_i \text{ for } 1 \leq i \leq n\}$ and $A^n \stackrel{\text{df}}{=} \{(a_1, a_2, \dots, a_n) \mid a_i \in A \text{ for } 1 \leq i \leq n\}$. We write $\{\varepsilon\} \stackrel{\text{df}}{=} A^0$ where ε is the *empty tuple* or *empty word*. Furthermore, we define the *Kleene plus* $A^+ \stackrel{\text{df}}{=} \bigcup_{i=1}^{\infty} A^i$ and the *Kleene star* (or *iteration*) $A^* \stackrel{\text{df}}{=} \bigcup_{i=0}^{\infty} A^i$. If A is a finite set then $|A|$ denotes its *cardinality*, the number of elements A contains. For a tuple $X = (a_1, a_2, \dots, a_n)$ we define $|X| \stackrel{\text{df}}{=} n$. Any non-empty finite set Σ is called an *alphabet* and its elements are called *symbols*. The elements of Σ^* are called *words*. If $w = (a_1, a_2, \dots, a_n)$ is a word, we also write $w = a_1 a_2 \dots a_n$ for $n \in \mathbb{N}$. The *reversal* of w is $w^R \stackrel{\text{df}}{=} a_n a_{n-1} \dots a_1$. For two words $w_1 = a_1 a_2 \dots a_n$ and $w_2 = b_1 b_2 \dots b_n$ we define the *concatenation* $w_1 \cdot w_2 \stackrel{\text{df}}{=} a_1 a_2 \dots a_n b_1 b_2 \dots b_n$ and we omit the dot most of the time. Furthermore, for $n \in \mathbb{N}$ we will sometimes identify $(w_1, w_2, \dots, w_n)$ and $w_1 w_2 \dots w_n$ for words w_i and $1 \leq i \leq n$ if it does not lead to confusion.

For an alphabet Σ the sets in $\mathcal{P}(\Sigma^*)$ are called *languages* over the alphabet Σ. For two languages L_1, L_2 we define $L_1 \cdot L_2 \stackrel{\text{df}}{=} \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$ and $L_1^R \stackrel{\text{df}}{=} \{w^R \mid w \in L_1\}$. An alphabet Σ is called *unary* if $|\Sigma| = 1$ and a language L is called *unary* if it is a language over a unary alphabet. We denote the *complementation* of a language L over the alphabet Σ by $\overline{L} \stackrel{\text{df}}{=} \Sigma^* - L$.

For a function (or mapping) $f \colon D \to W$ and sets $A \subseteq D$ we define $f(A) \stackrel{\text{df}}{=} \{f(x) \mid x \in A\}$ and for $B \subseteq W$ we define $f^{-1}(B) \stackrel{\text{df}}{=} \{x \in D \mid f(x) \in B\}$. For two functions f, g we define the *composition* as $f \circ g \colon x \mapsto f(g(x))$, if this is well-defined. A function $h \colon \Sigma^* \to \Delta^*$ for alphabets Σ, Δ is called a *homomorphism* if $h(uv) = h(u)h(v)$ for $u, v \in L^*$. h is called *ε-free* if $\varepsilon \notin h(\Sigma^+)$.

For a set A, a function $o \colon A^r \to A$, $r \in \mathbb{N}$ is called an *r-ary operation* on A (we say *unary* and *binary* for 1-ary and 2-ary, respectively). A set $C \subseteq A$ is called *closed under the operation o* if $o(C^r) \subseteq C$. For operations $o_1, o_2, \dots, o_n$ on A and a set $B \subseteq A$ we define the *algebraic closure of B under $o_1, o_2, \dots, o_n$* as $\Gamma_{o_1, o_2, \dots, o_n}(B) \stackrel{\text{df}}{=} \bigcap \{C \subseteq A \mid B \subseteq C,\ C \text{ is closed under } o_1, o_2, \dots, o_n\}$.

Examples of such operations are $\cup$ (union, binary), $\cap$ (intersection, binary), * (Kleene star, unary) and others.

For $r \in \mathbb{N}$ we call a set $\mathcal{O}$ of r-ary Operations on the same set A an *operation-aggregation* and call a set $B \subseteq A$ *closed under* $\mathcal{O}$ if B is closed under all operations in $\mathcal{O}$. We also use $\mathcal{O}$ as index of the closure operator Γ as shorthand for the sequence of all operations in $\mathcal{O}$. We additionally define the function $\mathcal{O}\colon \mathcal{P}(A) \to \mathcal{P}(A)$, $B \mapsto \bigcup_{o\in\mathcal{O}} o(B^r)$. Examples of operation-aggregations we will use in this thesis are:

- $h \stackrel{\text{df}}{=} \{g \mid g \text{ homomorphism}\}$ (*homomorphisms*)
- $\varepsilon h \stackrel{\text{df}}{=} \{g \mid g \ \varepsilon\text{-free homomorphism}\}$ (*ε-free homomorphisms*)
- $h^{-1} \stackrel{\text{df}}{=} \{g^{-1} \mid g \text{ homomorphism}\}$ (*inverse homomorphisms*)
- $\cap\text{REG} \stackrel{\text{df}}{=} \{L \mapsto L \cap R \mid R \in \text{REG}\}$ (*intersection with regular languages*)

Additionally, we define $\mathcal{M} \wedge \mathcal{K} \stackrel{\text{df}}{=} \{M \cap L \mid M \in \mathcal{M}, L \in \mathcal{K}\}$ for sets of languages $\mathcal{M}, \mathcal{K}$.

For a set A and $r \in \mathbb{N}$, a set $R \subseteq A^r$ is called an *r-ary relation on A*. R is *reflexive* if for all $x \in A$, $(x, x) \in R$ and *transitive* if $(x, y), (y, z) \in R$ implies $(x, z) \in R$ for all x, y, z. The *reflexive-transitive closure* of a relation $R \subseteq A^r$ is defined as $\bigcap\{S \subseteq A^r \mid R \subseteq S,\ S \text{ is reflexive and transitive}\}$.

In the next section, we will define several formal language classes. Other classes we will use in this thesis include P (the languages decidable in polynomial time), NP (the languages recognizable by nondeterministic machines in polynomial time) and RE (the recursively enumerable languages).

2.2 Formal Grammars

An introduction to the theory of formal languages can be found in [HMU01]. If not otherwise stated, the notions and definitions from that book will also be used here. Despite the fact that formal grammars are well-known, we will now cite some definitions and results so that we can directly refer to them in the rest of the thesis.

Definition (Formal grammar)**.**
The tuple $G = (\Sigma, N, N_0, P)$ is called a (formal) grammar *if*

- *Σ is an alphabet (the* terminals*),*
- *N is an alphabet disjoint from Σ (the* nonterminals*),*
- *$N_0 \in N$ (the* start symbol*) and*

- $P \subseteq (N \cup \Sigma)^+ \times (N \cup \Sigma)^*$ *(the* productions*) such that* P *is finite.*

For $(v, w) \in P$ *we also write* $v \to w$.

A word $y \in (\Sigma \cup N)^*$ *is* derivable in one step *from* $x \in (\Sigma \cup N)^*$ *by* G*, denoted as* $x \Rightarrow_G y$ *if there are* $u_1, v, w, u_2 \in (\Sigma \cup N)^*$ *such that* $v \to w \in P$ *and* $x = u_1 v u_2$ *and* $y = u_1 w u_2$*. The relation of* derivability*,* $\Rightarrow_G^*$*, is defined as the reflexive-transitive closure of* $\Rightarrow_G$.

The language generated by G *is defined as* $L(G) \stackrel{df}{=} \{w \in \Sigma^* \mid N_0 \Rightarrow_G^* w\}$.

Definition (Grammar types, formal languages)**.**
The formal grammar $G = (\Sigma, N, N_0, P)$ *is called*

- regular *if* $P \subseteq N \times (\Sigma \cdot (N \cup \{\varepsilon\}))$,
- context-free *if* $P \subseteq N \times (\Sigma \cup N)^*$ *and*
- context-sensitive *if for all* $(v, w) \in P$ *there exist* $v_1, u, v_2 \in (\Sigma \cup N)^*$, $|u| \geq 1$ *and* $A \in N$ *such that* $(v, w) = (v_1 A v_2, v_1 u v_2)$.

The respective language classes are defined in the following way:

- REG $\stackrel{df}{=} \{L \mid$ *there is a regular grammar* G *with* $L(G) = L - \{\varepsilon\}\}$
- CFL $\stackrel{df}{=} \{L(G) \mid G$ *is a context-free grammar*$\}$
- CSL $\stackrel{df}{=} \{L \mid$ *there is a context-sensitive grammar* G *with* $L(G) = L - \{\varepsilon\}\}$

The languages in REG*,* CFL *and* CSL *are also called* regular*,* context-free *and* context-sensitive*, respectively.*

From the restrictions on the productions, it is obvious that REG $\subseteq$ CFL, but it also holds that CFL $\subseteq$ CSL and both inclusions are strict. We want to illustrate this with some examples.

Example 2.1. *The language* $L_1 \stackrel{df}{=} \{a^n b^n \mid n \in \mathbb{N}\}$ *is context-free but not regular. It can be shown that the grammar* $G_1 \stackrel{df}{=} (\{a, b\}, \{S\}, S, \{S \to aSb, S \to \varepsilon\})$ *generates* L_1*, i.e.* $L(G_1) = L_1$*. Observe that* G_1 *is context-free, which implies that* $L_1 \in$ CFL*. The fact that* L_1 *is not regular can be shown using the pumping lemma for regular languages, which we will not do here.*

Example 2.2. *The language* $L_2 \stackrel{df}{=} \{a^n b^n c^n \mid n \in \mathbb{N}\}$ *is context-sensitive but not context-free. A context-sensitive grammar for* $L_2 - \{\varepsilon\}$ *can easily be found and it can be shown, using the pumping lemma for context-free languages, that* $L_2 \notin$ CFL*.*

We note that the class of context-free languages is the same as the class of languages that can be accepted by *pushdown automata* (PDAs). When one

restricts these automata to be deterministic (DPDA), the resulting class of accepted languages is called DCFL (*deterministic context-free languages*). As there are different non-equivalent methods to define DCFL, we choose the variant of deterministic state-accepting PDAs. We discuss some typical example languages.

Example 2.3. *The language* PAL $\stackrel{df}{=} \{w \in \{a,b\}^* \mid w = w^R\}$ *(the* palindromes*) is context-free but not deterministic context-free.* PAL *is context-free because* $G \stackrel{df}{=} (\{a,b\}, \{S\}, S, \{S \to aSa, S \to bSb, S \to a, S \to bS, \to \varepsilon\})$ *generates* PAL*. We will not prove that* PAL $\notin$ DCFL*.*

This example directly leads to the following classical result.

Proposition 2.4. DCFL $\subsetneq$ CFL

For a language quite similar to the palindromes, the inclusion in DCFL can be shown:

Example 2.5. *The language* PALS $\stackrel{df}{=} \{w\#w^R \mid w \in \{a,b\}^*\}$ *(the* palindromes with separation marker*) is deterministic context-free.*

Every context-free language can be generated by a context-free grammar in *Chomsky normal form*, which is sometimes useful for proofs. We will show that a normal form similar to this also exists for some recurrent circuits.

Definition (Chomsky normal form)**.**
A context-free grammar $G = (\Sigma, N, N_0, P)$ *is in* Chomsky normal form *if* $P \subseteq (N \times (\Sigma \cup N^2)) \cup \{N_0 \to \varepsilon\}$*.*

The following closure properties of the just defined language classes are well-known:

Proposition 2.6. *The class* REG *is closed under the operations* $\cup$, $\cap$, $^-$, $\cdot$, *, h, εh, h^{-1}, $\cap$REG *and* ^{R}*.*

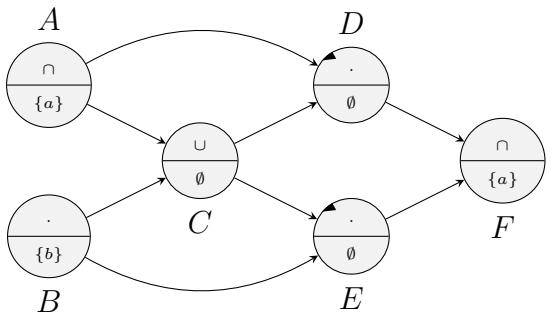
Proposition 2.7. *The class* CFL *is closed under the operations* $\cup$, $\cdot$, *, h, εh, h^{-1}, $\cap$REG *and* ^{R}*.* CFL *is neither closed under* $\cap$ *nor under* $^-$*.*

Corollary 2.8. CFL $\subsetneq \Gamma_\cap$(CFL)

Proof. This is obvious, since CFL is not closed under intersection (Proposition 2.7). □

3 Definition of Recurrent Circuits

We will give a short informal definition of recurrent circuits first. A recurrent circuit consists of gates that are connected via directed edges. The language generated in each gate is sent over all outgoing edges, and a language is generated in a gate by applying a fixed operation on all languages coming over ingoing edges. The circuits are not necessarily combinatoric, which means that circular paths via the edges are possible. Recurrent circuits do not have assigned input gates, in a way, every gate acts as an input gate because they are initialized with a certain value. We do not consider the reaction of the circuit if these input values are changed (i.e. the circuits are not interpreted as functions). An example of a recurrent circuit (though it does not really use recurrence) in graphical notation is given in Figure 1 together with the languages that are generated in each gate. The graphical notation will be explained later, but it should be quite intuitive.



$$L(A) = \{a\} \qquad L(B) = \{b\}$$
$$L(C) = L(A) \cup L(B) = \{a, b\} \qquad L(D) = L(A) \cdot L(C) = \{aa, ab\}$$
$$L(E) = L(C) \cdot L(B) = \{ab, bb\} \qquad L(F) = \{a\} \cup (L(D) \cap L(E)) = \{a, ab\}$$

Figure 1: Example circuit with generated languages L.

Now we will formally define the syntax of recurrent circuits in two steps. The fist step is to define the types of graphs that are allowed for the structure of such circuits.

Definition (Edge-ordered graph)**.**
The pair $G = (V, E)$ *is an* edge-ordered graph *if* V *is a finite set (the* nodes*) and* $E \colon \{1, 2, \ldots, n\} \to V \times V$ *for some* $n \in \mathbb{N}$*.*

An edge-ordered graph can be seen as a normal directed graph containing multi-edges with the additional property that all edges are linearly ordered. We will define some more notions about these graphs. So let $G = (V, E)$ be an edge-ordered graph with exactly $n \in \mathbb{N}$ edges. The elements of $\{1, 2, \ldots, n\}$ are called *edges* and if e is an edge such that $E(e) = (u, v)$, we say that e is an *edge from u to v*. The number of edges to a node v is called its *indegree*.

The edges must be linearly ordered, because an ordering on the ingoing edges is cruicial for a concatenation-gate (concatenation is the only non-commutative operation we consider). Since the order of the edges is actually only relevant between edges that have the same destination node, we will define the *ordered predecessor mapping* $\mathrm{pred}_G \colon V \to V^*$ for any $v \in V$ in the following way: $\mathrm{pred}_G(v) \stackrel{\mathrm{df}}{=} (u_1, u_2, \ldots, u_k)$ if there are edges $e_1 < e_2 < \cdots < e_k$ such that for any edge e and node u: $E(e) = (u, v) \iff \exists i \in \{1, 2, \ldots, k\}, e = e_i$ and $u = u_i$. This means that the tuple $\mathrm{pred}_G(v)$ contains exactly the predecessor nodes of v with multiplicities and in the correct edge-order. We will omit the subscript G if it does not lead to confusion. Note that the indegree of a node v ist exactly $|\mathrm{pred}_G(v)|$.

The second step in the definition of the syntax is the definition of the actual circuits.

Definition (Recurrent circuit)**.**
For a non-empty set $\mathcal{O} \subseteq \{\cup, \cap, \cdot\}$, the tuple $S = (\Sigma, V, V', E, \omega, \alpha)$ is called a recurrent $\mathcal{O}$-circuit *if*

- Σ *is an alphabet,*
- (V, E) *is an edge-ordered graph,*
- $V' \subseteq V$ *(the* output gates*),*
- $\omega \colon V \to \mathcal{O}$ *is a function (the* operations in the gates*) and*
- $\alpha \colon V \to \{A \subseteq \Sigma^* \mid |A| \leq 1\}$ *(the* initial languages*).*

We will use the names gate *and* node *for elements of V interchangeably. The definition of the predecessor mapping is extended to* $\mathrm{pred}_S \stackrel{df}{=} \mathrm{pred}_{(V,E)}$.

Next, we define the semantics of recurrent circuits.

Definition (Languages generated by recurrent circuits)**.**
Let $S = (\Sigma, V, V', E, \omega, \alpha)$ be a recurrent $\mathcal{O}$-circuit.

The set $S(v, t)$ generated at gate $v \in V$ with $\mathrm{pred}(v) = (u_1, u_2, \ldots, u_k)$ *after $t \in \mathbb{N}$ recursive steps is defined inductively as follows.*

- $S(v, 0) \stackrel{df}{=} \alpha(v)$

- *If* $k = 0$ *then* $S(v, t+1) \stackrel{df}{=} S(v,t)$ *and otherwise,*

$$S(v,t+1) \stackrel{df}{=} S(v,t) \cup \begin{cases} \bigcup_{i=1}^{k} S(u_k,t), & \text{if } \omega(v) = \cup \\ \bigcap_{i=1}^{k} S(u_k,t), & \text{if } \omega(v) = \cap \\ S(u_1,t) \cdot S(u_2,t) \cdot \ldots \cdot S(u_k,t), & \text{if } \omega(v) = \cdot \end{cases}$$

The set $S(v) \stackrel{df}{=} \bigcup_{t \geq 0} S(v,t)$ *is the* language generated by S at gate v *and* $L(S) \stackrel{df}{=} \bigcup_{v \in V'} S(v)$ *is the* language generated by S. *Two recurrent circuits* S, C *are* equivalent *if* $L(S) = L(C)$.

Note that for any $\mathcal{O} \subseteq \{\cup, \cap, \cdot\}$ and any node v of a recurrent $\mathcal{O}$-circuit S we have $S(v,0) \subseteq S(v,1) \subseteq S(v,2) \subseteq \cdots \subseteq S(v)$. Observe that for the semantics of a recurrent circuit it suffices to specify $\text{pred}(v)$ for any node v instead of the whole edge-mapping. Furthermore, for gates that are not concatenation-gates, a set of predecessors is already enough, so we will sometimes interpret the values of pred as sets and not as tuples.

When we restrict the set of available operations, we obtain different classes of languages which are the object of our study.

Definition (Recurrent circuit classes $\text{RC}(\mathcal{O})$)**.**
For a non-empty set of operations $\mathcal{O} \subseteq \{\cup, \cap, \cdot\}$*, we define the language class* $\text{RC}(\mathcal{O}) \stackrel{df}{=} \{L \mid L \text{ is generated by some } \mathcal{O}\text{-circuit}\}$.

For $o_1, o_2, \ldots, o_r \in \{\cup, \cap, \cdot\}$ we will also write $\text{RC}(o_1, o_2, \ldots, o_r)$ instead of $\text{RC}(\{o_1, o_2, \ldots, o_r\})$. Note that trivially, if $\emptyset \neq \mathcal{O}_1 \subseteq \mathcal{O}_2 \subseteq \{\cup, \cap, \cdot\}$, then also $\text{RC}(\mathcal{O}_1) \subseteq \text{RC}(\mathcal{O}_2)$.

For the sake of simplicity, we sometimes specify recurrent circuits in a graphical notation: Nodes are given as circles and have a double border if and only if they are output gates. Inside of the circles, the operation and the initial language are given in the top and the bottom half, respectively. Arrows denote edges between the gates. We specify the ordered predecessors of a concatenation-gate by a small arrowhead inside of the circle in the following way: The ingoing edges are always ordered either clockwise or counter-clockwise while the arrow indicates the first edge and the direction.

We will show how the language $0 \cdot \{0,1\}^* \cap \{0,1\}^* \cdot 0$ (the one given in the introduction) can be generated by a recurrent circuit and we will especially see how the iteration can be simulated by a concatenation-gate. This simulation will be discussed in detail later. Note that the given circuit is not minimal.

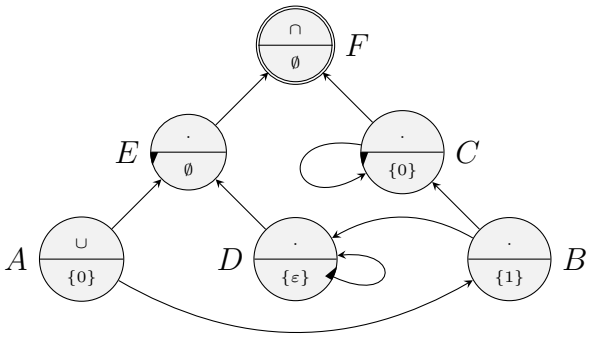


Figure 2: Example circuit that generates the language $0 \cdot \{0,1\}^* \cap \{0,1\}^* \cdot 0$.

Example 3.1. *We consider the circuit S given in Figure 2 node by node.*

The operation in node A is irrelevant, since A has no ingoing edges. Obviously, $S(A,0) = \alpha(A) = \{0\}$ and since A does not have predecessor gates, we get $S(A) = S(A,t) = \{0\}$ for any $t \in \mathbb{N}$.

For gate B we get $S(B,0) = \{1\}$ and $S(B,t+1) = S(B,t) \cup S(A,t) = \{0,1\}$ for any $t \in \mathbb{N}$ and thus $S(B) = \{0,1\}$. Note that the concatenation of only one language does not modify the language.

The gate C is a bit more complicated. Observe that the small arrowhead indicates that $\mathrm{pred}(C) = (C,B)$*. So we get $S(C,0) = \alpha(C) = \{0\}$ and for any $t \in \mathbb{N}$: $S(C,t+1) = S(C,t) \cup S(C,t) \cdot S(B,t) = S(C,t) \cup S(C,t) \cdot \{0,1\}$. Thus one can conclude that $S(C) = \{0\} \cdot \{0,1\}^*$.*

Similarly, for gate D we get $S(D,0) = \alpha(D) = \{\varepsilon\}$ and for any $t \in \mathbb{N}$: $S(D,t+1) = S(D,t) \cup S(D,t) \cdot S(B,t) = S(D,t) \cup S(D,t) \cdot \{0,1\}$ and thus $S(D,t) = \bigcup_{k=0}^{t} \{0,1\}^k$, which implies $S(D) = \{0,1\}^*$.

For gate E we have $\mathrm{pred}(E) = (A,D)$*, so we get $S(E,0) = \emptyset$ and $S(E,t+1) = S(E,t) \cup S(A,t) \cdot S(D,t) = S(E,t) \cup \{0\} \cdot \bigcup_{k=0}^{t} \{0,1\}^k$ for any $t \in \mathbb{N}$, which means that $S(E) = \{0\} \cdot \{0,1\}^*$.*

A more detailed analysis is needed for gate F. Here we have $S(F,0) = \emptyset$ and $S(F,t+1) = S(F,t) \cup (S(E,t) \cap S(C,t))$ for any $t \in \mathbb{N}$. Since $\alpha(F) = \emptyset$, we get for any $w \in \{0,1\}^*$ that $w \in S(F)$ if and only if there is a $t \in \mathbb{N}$ such that $w \in S(E,t)$ and $w \in S(C,t)$. Since the sets $S(E,t)$ and $S(C,t)$ are increasing in t and $S(E) = \bigcup_{t \in \mathbb{N}} S(E,t)$ and $S(C) = \bigcup_{t \in \mathbb{N}} S(C,t)$, we have $w \in S(F)$ if and only if $w \in S(E)$ and $w \in S(C)$. Then we get $S(F) = S(E) \cap S(C) = \{0\} \cdot \{0,1\}^* \cap \{0,1\}^* \cdot \{0\} = L(S)$ as we have claimed.

4 General Properties of Recurrent Circuits

4.1 Structural Properties

The first lemma will show that the concept of steps used to define recurrent circuits can be loosened a bit. We will see that it does not matter in which step a word is available in a gate, it only matters that it will be available at some step at all. This also means that one cannot exploit for example the fact that some words are produced in two gates always at the same step. One can actually insert arbitrary (also non-constant) delays without changing the semantics of the circuit. This quite natural property further justifies the choice of this particular definition for recurrent circuits.

Lemma 4.1. *Let $S = (\Sigma, V, V', E, \omega, \alpha)$ be a recurrent circuit and $v \in V$. If $|\mathrm{pred}(v)| = 0$, then $S(v) = \alpha(v)$ and if $\mathrm{pred}(v) = (u_1, u_2, \ldots, u_n)$ for $n \geq 1$, then*

$$S(v) = \alpha(v) \cup \begin{cases} \bigcup_{i=1}^{n} S(u_i), & \text{if } \omega(v) = \cup \\ \bigcap_{i=1}^{n} S(u_i), & \text{if } \omega(v) = \cap \\ (S(u_1) \cdot \ldots \cdot S(u_n)), & \text{if } \omega(v) = \cdot \end{cases}$$

Proof. Let $S = (\Sigma, V, V', E, \omega, \alpha)$ be a recurrent circuit and $v \in V$. If v does not have predecessors, the assertion is obvious, so let $\mathrm{pred}(v) = (u_1, u_2, \ldots, u_n)$.

For $\omega(v) = \cup$ we have

$$\begin{aligned} S(v) &= \alpha(v) \cup \bigcup_{t \geq 1} S(v, t) \\ &= \alpha(v) \cup \bigcup_{t \geq 1} \bigcup_{i=1}^{n} S(u_i, t-1) \\ &= \alpha(v) \cup \bigcup_{i=1}^{n} \bigcup_{t \geq 1} S(u_i, t-1) \\ &= \alpha(v) \cup \bigcup_{i=1}^{n} S(u_i) \end{aligned}$$

If $\omega(v) = \cap$ we get

$$\begin{aligned} S(v) &= \alpha(v) \cup \bigcup_{t \geq 1} S(v,t) \\ &= \alpha(v) \cup \bigcup_{t \geq 1} \bigcap_{i=1}^{n} S(u_i, t-1) \\ &= \alpha(v) \cup \bigcap_{i=1}^{n} \bigcup_{t \geq 1} S(u_i, t-1) \\ &= \alpha(v) \cup \bigcap_{i=1}^{n} S(u_i) \end{aligned}$$

The exchange of the infinite union and the finite intersection is legitimate because the sets $S(u_i, t)$ are increasing in t for any $u \in V$.

For $\omega(v) = \cdot$ we first show the inclusion "$\subseteq$":

$$\begin{aligned} S(v) &= \alpha(v) \cup \bigcup_{t \geq 1} S(v,t) \\ &= \alpha(v) \cup \bigcup_{t \geq 1} S(u_1, t-1) \cdot S(u_2, t-1) \cdot \ldots \cdot S(u_n, t-1) \\ &\subseteq \alpha(v) \cup \bigcup_{t \geq 1} S(u_1) \cdot S(u_2) \cdot \ldots \cdot S(u_n) \\ &= \alpha(v) \cup S(u_1) \cdot S(u_2) \cdot \ldots \cdot S(u_) \end{aligned}$$

For "$\supseteq$" let $w \in \alpha(v) \cup S(u_1) \cdot S(u_2) \cdot \ldots \cdot S(u_n)$. If $w \in \alpha(v)$ then we have $w \in S(v,0) \subseteq S(v)$.
Otherwise, there is a $t \geq 0$ such that $w \in S(u_1, t) \cdot S(u_2, t) \cdot \ldots \cdot S(u_n, t) \subseteq S(v, t+1) \subseteq S(v)$. □

This lemma also suggests the connection between recurrent circuits and language equations. We will talk about language equations later, but we want to note that there is a close correspondence between recurrent circuits, language equations and formal grammars. One can directly transform each of the three into another one. Later we will show the equivalence of some kinds of recurrent circuits with types of formal grammars (using language equations). Formal grammars and recurrent circuits both use the concept of steps to generate words. This concept is not necessary for language equations and the

preceding lemma clarifies why one can also define these classes without the need of steps.

A normal form similar to the Chomsky normal form also exists for recurrent circuits. We will prove this step-by-stey in the following.

Lemma 4.2. *If $L \in \mathrm{RC}(\mathcal{O})$ for $\mathcal{O} \subseteq \{\cup, \cap, \cdot\}$ then there is an $\mathcal{O}$-circuit C such that $L = L(C)$ and each gate in C has indegree at most two.*

Proof. Let $C = (\Sigma, V, V', E, \omega, \alpha)$ be a recurrent circuit and $v \in V$ be a gate such that $\mathrm{pred}(v) = (v_1, v_2, \ldots, v_n)$, $n \geq 3$ and $\circ \stackrel{\mathrm{df}}{=} \omega(v) \in \{\cup, \cap, \cdot\}$. We will construct a circuit $C' \stackrel{\mathrm{df}}{=} (\Sigma, V \cup \{x, y\}, V', E', \omega', \alpha')$, where x, y are new gates, such that $L(C) = L(C')$, x and v have two predecessors in C' and y has strictly less predecessors in C' than v in C. When this construction is iterated, one obtains a circuit that has the desired properties. So let us finish the definition of C'. We implicitly specify E' by defining $\mathrm{pred}_{C'}$. $\mathrm{pred}_{C'}(v) \stackrel{\mathrm{df}}{=} (x, y)$, $\mathrm{pred}_{C'}(x) \stackrel{\mathrm{df}}{=} (v_1, v_2)$, $\mathrm{pred}_{C'}(y) \stackrel{\mathrm{df}}{=} (v_3, v_4, \ldots, v_n)$ and $\mathrm{pred}_{C'}(u) \stackrel{\mathrm{df}}{=} \mathrm{pred}_C(u)$ for all $u \notin \{v, x, y\}$. For $u \in V$ we define $\alpha'(u) \stackrel{\mathrm{df}}{=} \alpha(u)$ and $\omega'(u) \stackrel{\mathrm{df}}{=} \omega(u)$ and $\alpha'(x) \stackrel{\mathrm{df}}{=} \alpha'(y) \stackrel{\mathrm{df}}{=} \emptyset$, and finally $\omega'(x) \stackrel{\mathrm{df}}{=} \omega'(y) \stackrel{\mathrm{df}}{=} \circ$.

We will now show the equivalence of these circuits.

Claim 4.2.1. *For all $u \in V$ and $t \in \mathbb{N}$ holds:*
$C'(u,t) \subseteq C(u)$, $C'(x,t) \subseteq C(v_1) \circ C(v_2)$ and $C'(y,t) \subseteq C(v_3) \circ \cdots \circ C(v_n)$.

Proof of the claim. We prove this by induction on $t \in \mathbb{N}$.

Induction basis. We get $C'(v, 0) = \alpha'(v) = \alpha(v) \subseteq C(v)$, the assertion is obvious for the other gates.

Induction step. Let $t \in \mathbb{N}$. The induction step is trivial for $u \notin \{v, x, y\}$. Using Lemma 4.1 and the associativity of $\circ$, we get for the other gates:

$$\begin{aligned}
C'(v, t+1) &= C'(v,t) \cup (C'(x,t) \circ C'(y,t)) \\
&\stackrel{\mathrm{IH}}{\subseteq} C(v) \cup (C(v_1) \circ C(v_2) \circ C(v_3) \circ \cdots \circ C(v_n)) \\
&\subseteq C(v), \\
C'(x, t+1) &= C'(x,t) \cup (C'(v_1,t) \circ C'(v_2,t)) \\
&\stackrel{\mathrm{IH}}{\subseteq} C(v_1) \circ C(v_2), \\
C'(y, t+1) &= C'(y,t) \cup (C'(v_3,t) \circ C'(v_4,t) \circ \cdots \circ C'(v_n,t)) \\
&\stackrel{\mathrm{IH}}{\subseteq} C(v_3) \circ C(v_4) \circ \cdots \circ C(v_n).
\end{aligned}$$

□

Claim 4.2.2. *$C(u,t) \subseteq C'(u)$ for all $u \in V$ and $t \in \mathbb{N}$.*

Proof of the claim. We again prove this by induction on $t \in \mathbb{N}$.

Induction basis. We only need to consider the gate v: $C(v,0) = \alpha(v) = \alpha'(v) \subseteq C'(v)$.

Induction step. Let $t \in \mathbb{N}$. It suffices to show the induction step for the gate v. Again, we use Lemma 4.1 and the associativity of $\circ$:

$$\begin{aligned} C(v,t+1) &= C(v,t) \cup (C(v_1,t) \circ C(v_2,t) \circ \cdots \circ C(v_n,t)) \\ &\overset{\text{IH}}{\subseteq} C'(v) \cup (C'(v_1) \circ C'(v_2) \circ \cdots \circ C'(v_n)) \\ &\subseteq C'(v) \cup (C'(x) \circ C'(y)) \\ &\subseteq C'(v) \end{aligned}$$

□

Using both claims together we get $C(u) = \bigcup_{t\in\mathbb{N}} C(u,t) \subseteq C'(u)$ and $C'(u) = \bigcup_{t\in\mathbb{N}} C'(u,t) \subseteq C(u)$ for any $u \in V$ and thus $L(C) = L(C')$. □

Lemma 4.3. *If $L \in \mathrm{RC}(\mathcal{O})$ for $\mathcal{O} \subseteq \{\cup, \cap, \cdot\}$ then there is an $\mathcal{O}$-circuit C such that $L = L(C)$ and every gate in C has indegree exactly two.*

Proof. Let C be an $\mathcal{O}$-circuit such that $L \in \mathrm{RC}(C)$. By Lemma 4.2, we can assume that this circuit is constructed in a way such that every node has indegree at most two. We now want to transform C into a circuit C' that has the asserted properties. To this end, we add a gate e_1 with $\emptyset$ as initial set and any operation from $\mathcal{O}$. If $\cdot \in \mathcal{O}$, we also add a $\cdot$-gate e_2 with initial set $\{\varepsilon\}$. Both e_1 and e_2 each have two ingoing edges coming from themselves. Note that $C(e_1,t) = \emptyset$ and $C(e_2) = \{\varepsilon\}$ for every $t \in \mathbb{N}$.

For each gate with indegree zero, we add two edges from e_1. For $\cup$- and $\cap$-gates that have exactly one ingoing edge, we simply repeat this edge. Finally, $\cdot$-gates with indegree one get an additional edge from e_2.

Observe that the circuit has the required structure and since the additional edges from e_1 and e_2 only apply the neutral element of the respective operation, it is obvious that the so-obtained circuit C' also generates L. □

For some sets of operations, we can improve the result from Lemma 4.3 so that we get a normal form that further resembles the Chomsky normal form for context-free grammars. The construction works exactly as in the context-free case and has already been suggested by Okhotin [Okh01].

Theorem 4.4 (Binary Normal Form)**.**
Let $\{\cup, \cdot\} \subseteq \mathcal{O} \subseteq \{\cup, \cap, \cdot\}$*. For any* $L \in \mathrm{RC}(\mathcal{O})$ *there exists a recurrent* $\mathcal{O}$*-circuit* $C = (\Sigma, V, V', E, \omega, \alpha)$ *such that* $L = L(C)$*, every gate in* C *has indegree exactly two,* $V' = \{S\}$*,* S *does not have outgoing edges and for every* $v \in V - V'$*,* $\alpha(v) \subseteq \Sigma$*.*

Proof. Let $C = (\Sigma, V, V', E, \omega, \alpha)$ be an $\mathcal{O}$-circuit such that $L(C) = L$. For any $w \in \Sigma^+$, the language $\{w\}$ can obviously be generated by a $\{\cdot\}$-circuit C_w, such that for every node v in C_w, $\alpha(v) = \{a\}$ for some $a \in \Sigma$. Then, nodes v in C that have initial sets containing words which are longer than one symbol can be replaced by a subcircuit that uses a $\cup$-gate and the circuit C_w such that the initial set of v can be the empty set. This circuit obviously generates the same language as C and does not have words in initial sets that are longer than one symbol. So let us assume that C has this property. Furthermore, we can also assume that every gate in C has indegree exactly two by Lemma 4.3 and this assumption is independent of the assumption about the initial sets.

Let now $C_1 = (\Sigma, V \cup \{S\}, \{S\}, E_1, \omega_1, \alpha_1)$ be the modified circuit that has one additional $\cup$-gate S which is its only output gate, $\mathrm{pred}(S) \stackrel{\mathrm{df}}{=} V'$ and $\alpha(S) = \emptyset$. Then we obviously have $L(C_1) = \bigcup_{v \in V'} C_1(v) = \bigcup_{v \in V'} C(v) = L(C)$, by Lemma 4.1. We can again assume that all the gates in C_1 have exactly two predecessors. Note that the properties of C_1 about the output gate do not change by this modification.

It remains to show that we can eliminate initial sets of the form $\{\varepsilon\}$ in each gate that is not the output gate. To this end, we construct the circuit $C_2 = (\Sigma, V_2, \{S\}, E_2, \omega_2, \alpha_2)$ by replacing some gates in C_1 and removing the empty word from the initial sets:

For every $\cdot$-gate v in C_1 with $\mathrm{pred}(v) = (p_1, p_2)$, we replace v by a $\cup$-gate v' and introduce a new $\cdot$-gate x such that (cf. Figure 3):

- $\omega_2(v') \stackrel{\mathrm{df}}{=} \cup$, $\omega_2(x) \stackrel{\mathrm{df}}{=} \cdot$, $\alpha_2(v') \stackrel{\mathrm{df}}{=} \alpha_1(v) - \{\varepsilon\}$, $\alpha(x) \stackrel{\mathrm{df}}{=} \emptyset$
- All outgoing edges of v are now outgoing edges of v'.
- $\mathrm{pred}(x) \stackrel{\mathrm{df}}{=} (p_1, p_2)$
- $\{x\} \subseteq \mathrm{pred}(v') \subseteq \{x, p_1, p_2\}$ such that
  - $p_1 \in \mathrm{pred}(v') \iff \varepsilon \in C_1(p_2)$
  - $p_2 \in \mathrm{pred}(v') \iff \varepsilon \in C_1(p_1)$

Additionally, for every other gate $v \in V_1 - \{S\}$ and its corresponding gate $v' \in V_2$, we set $\alpha_2(v') \stackrel{\mathrm{df}}{=} \alpha_1(v) - \{\varepsilon\}$. Finally, if $\varepsilon \in L(C_1)$, then $\alpha_2(S) \stackrel{\mathrm{df}}{=} \{\varepsilon\}$ and otherwise $\alpha_2(S) \stackrel{\mathrm{df}}{=} \emptyset$.

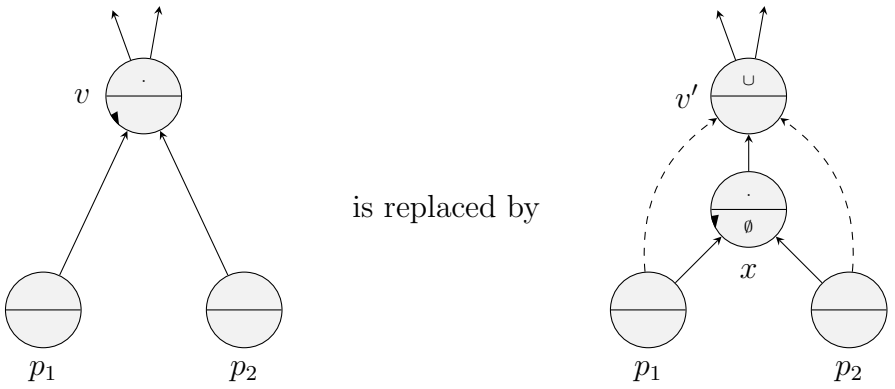


Figure 3: Replacement scheme used in the proof of Theorem 4.4.

Note that only the newly introduced $\cup$-gates v' in the circuit C_2 can have more or less than two ordered predecssors, but of course a circuit with only small modifications can be constructed that fulfills the structural requirements of the theorem.

For the rest of the proof, we define $X_\varepsilon \stackrel{\text{df}}{=} X - \{\varepsilon\}$ for a language X.

Claim 4.4.1. *For each gate $v \in V$ in C_1 and its corresponding gate v' in C_2 and every $t \in \mathbb{N}$:*

$$C_1(v,t)_\varepsilon \subseteq C_2(v')$$

Proof of the claim. Induction basis. Since we have only removed the empty word from initial sets (if at all), the case $t = 0$ is clear.

Induction step. Note that we only have to argue for $\cdot$-gates v_1 in C_1. So let v_2 be the corresponding $\cup$-gate in C_2 and x be the new $\cdot$-gate in C_2. By p_1 and p_2 we will denote the ordered predecessors of both v_1 and x. Using Lemma 4.1, we obtain for $t \in \mathbb{N}$:

$$\begin{aligned} C_1(v_1,t+1)_\varepsilon &= C_1(v_1,t)_\varepsilon \cup (C_1(p_1,t)C_1(p_2,t))_\varepsilon \\ &\stackrel{\text{IH}}{\subseteq} C_2(v_2) \cup (C_1(p_1,t)C_1(p_2,t))_\varepsilon \end{aligned}$$

So it remains to show that $(C_1(p_1,t)C_1(p_2,t))_\varepsilon \subseteq C_2(v_2)$.

Case 1, $\varepsilon \notin C_1(p_1) \cup C_2(p_2)$:

$$\begin{aligned} (C_1(p_1,t)C_1(p_2,t))_\varepsilon &= C_1(p_1,t)_\varepsilon C_1(p_2,t)_\varepsilon \stackrel{\text{IH}}{\subseteq} C_2(p_1)C_2(p_2) \\ &= C_2(x) \subseteq C_2(v_2). \end{aligned}$$

Case 2, $\varepsilon \in C_1(p_1) - C_1(p_2)$:

$$\begin{aligned}(C_1(p_1,t)C_1(p_2,t))_\varepsilon &\subseteq C_1(p_1,t)_\varepsilon C_1(p_2,t)_\varepsilon \cup C_1(p_2,t)_\varepsilon \\ &\overset{\text{IH}}{\subseteq} C_2(p_1)C_2(p_2) \cup C_2(p_2) \\ &\subseteq C_2(x) \cup C_2(v_2) \subseteq C_2(v_2)\end{aligned}$$

Case 3, $\varepsilon \in C_1(p_2) - C_1(p_1)$: see case 2.

Case 4, $\varepsilon \in C_1(p_1) \cap C_2(p_2)$:

$$\begin{aligned}(C_1(p_1,t)C_1(p_2,t))_\varepsilon &\subseteq C_1(p_1,t)_\varepsilon C_1(p_2,t)_\varepsilon \cup C_1(p_1,t)_\varepsilon \cup C_1(p_2,t)_\varepsilon \\ &\overset{\text{IH}}{\subseteq} C_2(p_1)C_2(p_2) \cup C_2(p_1) \cup C_2(p_2) \\ &\subseteq C_2(x) \cup C_2(v_2) \cup C_2(v_2) \subseteq C_2(v_2)\end{aligned}$$

This completes the induction. □

This claim actually implies $C_1(v)_\varepsilon \subseteq C_2(v')$ for any gate $v \neq S$ in C_1 and its corresponding gate v' in C_2. We now show the other direction with a similar claim.

Claim 4.4.2. *For each gate $v \in V$ in C_1 and its corresponding gate v' in C_2 and every $t \in \mathbb{N}$:*

$$C_2(v',t) \subseteq C_1(v)_\varepsilon$$

Proof of the claim. Again, it suffices to argue for $\cdot$-gates v_1 of the circuit C_1. Let again v_2 be the corresponding $\cup$-gate and x be the $\cdot$-gate in C_2.

We use [and] to specify parts of expressions that occur only for some cases of $\varepsilon \in C_1(p_1), \varepsilon \in C_1(p_2)$. Observe that the statements are correct in any of the four cases.

Induction basis. The case $t = 0$ is clear since we have removed all the empty words from the initial sets. We also need the next step:

$$\begin{aligned}C_2(v_2,1) &= C_2(v_2,0) \cup C_2(x,0)\big[\cup C_2(p_1,0) \cup C_2(p_2,0)\big] \\ &= C_1(v_1,0)_\varepsilon \cup \emptyset\big[\cup C_1(p_1,0)_\varepsilon \cup C_1(p_2,0)_\varepsilon\big] \\ &\subseteq C_1(v_1)_\varepsilon\big[\cup C_1(p_1)_\varepsilon \cup C_1(p_2)_\varepsilon\big] \\ &\subseteq C_1(v_1)_\varepsilon\end{aligned}$$

Induction step. For any $t \in \mathbb{N}$ we have

$$\begin{aligned} C_2(v_2, t+2) &= C_2(v_2, t+1) \cup C_2(x, t+1)\big[\cup C_2(p_1, t+1) \cup C_2(p_2, t+1)\big] \\ &\overset{\text{IH}}{\subseteq} C_1(v_1)_\varepsilon \cup C_2(p_1, t)C_2(p_2, t)\big[\cup C_1(p_1)_\varepsilon \cup C_1(p_2)_\varepsilon\big] \\ &\overset{\text{IH}}{\subseteq} C_1(v_1)_\varepsilon \cup C_1(p_1)_\varepsilon C_1(p_1)_\varepsilon\big[\cup C_1(p_1)_\varepsilon \cup C_1(p_2)_\varepsilon\big] \\ &\subseteq C_1(v_1)_\varepsilon. \end{aligned}$$

The inclusion $C_1(p_1)_\varepsilon \subseteq C_1(v_1)_\varepsilon$ holds because when the expression $C_1(p_1)_\varepsilon$ is present, $\varepsilon \in C_1(p_2)$ and thus $C_1(p_1) \subseteq C_1(v_1)$ and analogously for the other cases. This completes the induction. □

These two claims together imply for any gate $v \neq S$ in C_1 and the corresponding gate v' in C_2:

$$C_1(v)_\varepsilon = \bigcup_{t \in \mathbb{N}} C_1(v, t)_\varepsilon = \bigcup_{t \in \mathbb{N}} C_2(v', t) = C_2(v')$$

Since the output gate S does not have outgoing edges, and $\varepsilon \in \alpha_2(S) \iff \varepsilon \in L(C_1)$, this means that $L(C_1) = L(C_2)$ and thus the proof is complete. □

Observe that a circuit C of the form as in the previous theorem has the property that $\varepsilon \notin C(v)$ for every gate v which is not the output gate. This property can be used by inductions over the length of words generated by gates of such a circuit, since if a word w, $|w| \geq 2$ is produced at some $\cdot$-gate for the first time, this ensures that there are shorter words w_1 and w_2, which are generated in predecessor gates and $w_1 w_2 = w$.

4.2 Closure Properties

In the following, we show some closure properties for recurrent-circuit languages. Since each circuit can have multiple output gates, the fact that every recurrent circuit class is closed under union is not very surprising.

Proposition 4.5. *For any $\mathcal{O} \subseteq \{\cup, \cap, \cdot\}$ the class $\mathrm{RC}(\mathcal{O})$ is closed under union.*

Proof. Let $\mathcal{O} \subseteq \{\cup, \cap, \cdot\}$, $L_i \in \mathrm{RC}(\mathcal{O})$ and $L_i = L(C_i)$ for some recurrent $\mathcal{O}$-circuit $C_i = (\Sigma, V_i, V_i', E_i, \omega_i, \alpha_i)$ with exactly n_i edges for $i = 1, 2$. We assume that V_1 and V_2 are disjoint.

A circuit for $L_1 \cup L_2$ is simply the combination of C_1 and C_2 into one (unconnected) circuit, namely $C \stackrel{\mathrm{df}}{=} (\Sigma, V_1 \cup V_2, V_1' \cup V_2', E, \omega, \alpha)$, where $E(i) \stackrel{\mathrm{df}}{=} E_1(i)$ for $1 \leq i \leq n_1$ and $E(n_1 + i) \stackrel{\mathrm{df}}{=} E_2(i)$ for $1 \leq i \leq n_2$. Furthermore, ω is the common extension of ω_1 and ω_2 to the domain $V_1 \cup V_2$ and α likewise. Obviously, $L(C) = L(C_1) \cup L(C_2)$. Note that we have not added new gates and that C is again an $\mathcal{O}$-circuit. □

It is generally the case that a class of languages generated by recurrent circuits with certain operations is closed under all these operations. A minor problem arises here, since recurrent circuits can have multiple output gates, but this can be resolved, as we will see next.

Proposition 4.6. *For every $\mathcal{O} \subseteq \{\cup, \cap, \cdot\}$, the class $\mathrm{RC}(\mathcal{O})$ is closed under the operations in $\mathcal{O}$.*

Proof. The closure under union has already been shown in Proposition 4.5.

Let $\mathcal{O} \subseteq \{\cup, \cap, \cdot\}$, $L_i \in \mathrm{RC}(\mathcal{O})$ and $C_i = (\Sigma, V_i, V_i', E_i, \omega_i, \alpha_i)$ a recurrent $\mathcal{O}$-circuit such that $L(C_i) = L_i$ and $n_i \stackrel{\mathrm{df}}{=} |V_i|$ for $i = 1, 2$. Furthermore, we assume that V_1 and V_2 are disjoint.

We argue simultaneously for the closure under intersection and under concatenation, so let $\circ = \cap$ or $\circ = \cdot$, respectively. We can assume that each of the circuits C_1 and C_2 has at least one output gate. We construct the circuit C that generates $L(C_1) \circ L(C_2)$ in the following way.

$C \stackrel{\mathrm{df}}{=} (\Sigma, V, V', E, \omega, \alpha)$, where $V' \stackrel{\mathrm{df}}{=} \{x_{u,v} \mid u \in V_1', v \in V_2'\}$ are new gates, $V \stackrel{\mathrm{df}}{=} V_1 \cup V_2 \cup V'$, $\omega(v) \stackrel{\mathrm{df}}{=} \begin{cases} \omega_1(v) & \text{for } v \in V_1 \\ \omega_2(v) & \text{for } v \in V_2 \\ \circ & \text{for } v \in V' \end{cases}$, $\alpha(v) \stackrel{\mathrm{df}}{=} \begin{cases} \alpha_1(v) & \text{for } v \in V_1 \\ \alpha_2(v) & \text{for } v \in V_2 \text{ and} \\ \emptyset & \text{for } v \in V' \end{cases}$
$E(i) \stackrel{\mathrm{df}}{=} E_1(i)$ for $1 \leq i \leq n_1$, $E(n_1 + i) \stackrel{\mathrm{df}}{=} E_2(i)$ for $1 \leq i \leq n_2$, furthermore, edges are included in the circuit such that $\mathrm{pred}(x_{u,v}) = (u, v)$ for every $u \in V_1'$ and $v \in V_2'$.

Then we have, using Lemma 4.1,

$$L(C) = \bigcup_{u\in V_1', v\in V_2'} L(x_{u,v}) = \bigcup_{u\in V_1', v\in V_2'} (L(u) \circ L(v)) =$$
$$= \Big(\bigcup_{u\in V_1'} L(u)\Big) \circ \Big(\bigcup_{v\in V_2'} L(v)\Big) = L(C_1) \circ L(C_2) = L_1 \circ L_2.$$

And thus, $L_1 \circ L_2 \in \mathrm{RC}(\mathcal{O})$. □

We have already mentioned that the Kleene star can be simulated by a $\cdot$-gate. By explicitly constructing a gate that simulates the Kleene star, we will show the closure under iteration if the operation $\cdot$ is available.

Proposition 4.7. *For $\{\cdot\} \subseteq \mathcal{O} \subseteq \{\cup, \cap, \cdot\}$ the class $\mathrm{RC}(\mathcal{O})$ is closed under iteration.*

Proof. Let $\{\cdot\} \subseteq \mathcal{O} \subseteq \{\cup, \cap, \cdot\}$ and $C = (\Sigma, V, V', E, \omega, \alpha)$ be an $\mathcal{O}$-circuit such that $V' = \{v_1, v_2, \ldots, v_n\}$ and we assume $n \geq 1$.

We now construct the circuit C^* such that $L(C^*) = (L(C))^*$. For a schematic view, see Figure 4. $C^* \stackrel{\text{df}}{=} (\Sigma, V \cup \{s_1, s_2, \ldots, s_n, s\}, \{s\}, E^*, \omega^*, \alpha^*)$, where $s_1, s_2, \ldots, s_n, s$ are new gates and

$$\omega^*(v) \stackrel{\text{df}}{=} \begin{cases} \omega(v) & \text{if } v \in V \\ \cdot & \text{if } v \notin V \end{cases}, \qquad \alpha^*(v) \stackrel{\text{df}}{=} \begin{cases} \alpha(v) & \text{if } v \in V \\ \{\varepsilon\} & \text{if } v \notin V \end{cases},$$

$$\mathrm{pred}_{C^*}(v) \stackrel{\text{df}}{=} \begin{cases} \mathrm{pred}_C(v) & \text{if } v \in V \\ (v_i) & \text{if } v = s_i \text{ for some } i \in \{1, 2, \ldots, n\} \\ (s, v_1, v_2, \ldots, v_n) & \text{if } v = s \end{cases}$$

Note that the gate s simulates the iteration, which will be especially clear for $n = 1$. We now have to show that $(L(C))^* = (C(v_1) \cup C(v_2) \cup \cdots \cup C(v_n))^* = C^*(s) = L(C^*)$.

From Lemma 4.1 and the fact that the ingoing egdes for gates in C are not changed in C^* we obtain

$$\begin{aligned} C^*(s) &= \{\varepsilon\} \cup C^*(s) \cdot C^*(s_1) \cdot C^*(s_2) \cdot \ldots \cdot C^*(s_n) \\ &= \{\varepsilon\} \cup C^*(s) \cdot (C^*(v_1) \cup \{\varepsilon\}) \cdot (C^*(v_2) \cup \{\varepsilon\}) \cdot \ldots \cdot (C^*(v_n) \cup \{\varepsilon\}) \\ &= \{\varepsilon\} \cup C^*(s) \cdot (C(v_1) \cup \{\varepsilon\}) \cdot (C(v_2) \cup \{\varepsilon\}) \cdot \ldots \cdot (C(v_n) \cup \{\varepsilon\}) \end{aligned}$$

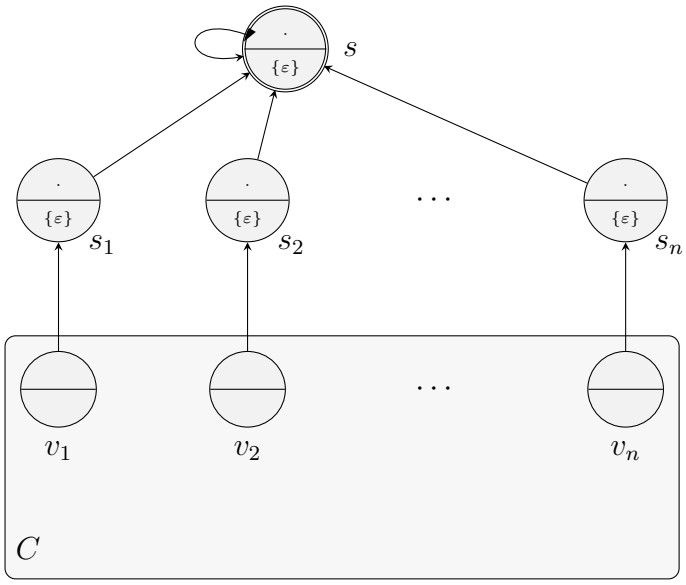


Figure 4: The circuit C^*, used in the proof of Proposition 4.7.

and now show $(L(C))^k \subseteq C^*(s)$ by induction over $k \in \mathbb{N}$.

Induction basis. $(L(C))^0 = \{\varepsilon\} \subseteq C^*(s)$.

Induction step. Let $k \in \mathbb{N}$. Using again Lemma 4.1, we obtain

$$\begin{aligned}(L(C))^{k+1} &= (L(C))^k \cdot L(C) = (L(C))^k \cdot (C(v_1) \cup C(v_2) \cup \cdots \cup C(v_n)) \\ &\overset{\text{IH}}{\subseteq} C^*(s) \cdot (C(v_1) \cup C(v_2) \cup \cdots \cup C(v_n)) \\ &\subseteq C^*(s)\end{aligned}$$

by the above equality.

So we have shown $(L(C))^* \subseteq L(C^*)$ and it remains to show the other inclusion. For this, we prove $C^*(s,t) \subseteq (L(C))^*$ by induction over $t \in \mathbb{N}$.

Induction basis. We have $C^*(s,0) = \{\varepsilon\} \in (L(C))^*$ and furthermore also $C^*(s,1) = C^*(s,0) \cup C^*(s,0) \cdot C^*(s_1,0) \cdot \ldots \cdot C^*(s_n,0) = \{\varepsilon\} \in (L(C))^*$.

Induction step. Let $t \in \mathbb{N}$. From the definition it is clear that

$$\begin{aligned}C^*(s_i, t+1) &= \alpha(s_i) \cup \bigcup_{t'=0}^{t} (C^*(s_i, t') \cup C^*(v_i, t')) \\ &= \{\varepsilon\} \cup C^*(v_i, t) = \{\varepsilon\} \cup C(v_i, t)\end{aligned}$$

for any $i \in \{1, 2, \ldots, n\}$. So we can conclude

$$\begin{aligned} C^*(s, t+2) &= C^*(s, t+1) \cup C^*(s, t+1) \cdot C^*(s_1, t+1) \cdot \ldots \cdot C^*(s_n, t+1) \\ &\overset{\text{IH}}{\subseteq} (L(C))^* \cup (L(C))^* \cdot (C(v_1, t) \cup \{\varepsilon\}) \cdot \ldots \cdot (C(v_n, t) \cup \{\varepsilon\}) \\ &\subseteq (L(C))^* \cup (L(C))^* \cdot (C(v_1, t) \cup \cdots \cup C(v_n, t))^* \\ &\subseteq (L(C))^* \end{aligned}$$

Since we have shown that $L(C^*) = (L(C))^*$, there is a recurrent $\mathcal{O}$-circuit that generates $(L(C))^*$ and thus the assertion holds. □

As another closure property, we can show the closure under homomorphisms for most of the classes of recurrent circuits. We will use this result in some of the following proofs. Later, we will see that all other classes of recurrent circuits are not closed under homomorphisms.

Theorem 4.8. *For every $\mathcal{O} \subseteq \{\cup, \cdot\}$, the class $\mathrm{RC}(\mathcal{O})$ is closed under homomorphisms.*

Proof. Let $\varphi\colon \Sigma^* \to \Delta^*$ be a homomorphism and $C = (\Sigma, V, V', E, \omega, \alpha)$ a recurrent $\mathcal{O}$-circuit. We now argue that $C' = (\Sigma, V, V', E, \omega, \varphi \circ \alpha)$ is the circuit that accepts $\varphi(L(C))$.

We obviously have $\varphi(L(C)) = \bigcup_{v \in V'} \varphi(C(v)) = \bigcup_{v \in V'} \bigcup_{t \geq 0} \varphi(C(v, t))$. It now suffices to show by induction over $t \geq 0$ that $\varphi(C(v, t)) = \varphi(C'(v, t))$ for every $v \in V$.
Let $\mathrm{pred}(v) = (u_1, u_2, \ldots, u_n)$.
Induction basis. Here we get $\varphi(C(v, 0)) = \varphi(\alpha(v)) = C'(v, 0)$.

Induction step. Let now $t \geq 1$ and $v \in V$. If $\omega(v) = \cup$ then we have by definition:

$$\varphi(C(v, t)) = \varphi(C(v, t-1)) \cup \bigcup_{i=1}^{n} \varphi(C(u_i, t-1)) \overset{\text{IH}}{=} C'(v, t).$$

If $\omega(v) = \cdot$, we get:

$$\begin{aligned} \varphi(C(v, t)) &= \varphi(C(v, t-1)) \cup \varphi(C(u_1, t-1) \cdot \ldots \cdot C(u_n, t-1)) \\ &= \varphi(C(v, t-1)) \cup \varphi(C(u_1, t-1)) \cdot \ldots \cdot \varphi(C(u_n, t-1)) \\ &\overset{\text{IH}}{=} C'(v, t) \end{aligned}$$

□

We conclude this section with an immediate closure property.

Proposition 4.9. *For $\mathcal{O} \subseteq \{\cup, \cap, \cdot\}$ the class $\mathrm{RC}(\mathcal{O})$ is closed under reversal.*

Proof. Let $L \subseteq \Sigma^*$ be the language generated by the recurrent circuit $C = (\Sigma, V, V', E, \omega, \alpha)$ with exactly n gates. Obviously, L^R is generated by the circuit $C \stackrel{\mathrm{df}}{=} (\Sigma, V, V', E', \omega, \alpha')$ where $E'(i) \stackrel{\mathrm{df}}{=} E(n - i + 1)$ and $\alpha'(v) \stackrel{\mathrm{df}}{=} \alpha(v)^R$. □

5 Equalities to Known Classes

5.1 Circuits Without Concatenation

Not surprisingly, the classes of recurrent circuits that cannot use concatenation, namely RC($\cup$), RC($\cap$) and RC($\cup, \cap$), will turn out to be not very expressive. Nevertheless, the class of languages they represent is quite familiar, which we will see in this section.

Definition (Finite languages FIN)**.**
The language class FIN *of all finite languages over arbitrary alphabets is defined by* $\text{FIN} \stackrel{df}{=} \{L \mid L \text{ is a finite language}\}$.

Theorem 5.1. *For* $\mathcal{O} \subseteq \{\cup, \cap\}$ *it holds that* $\text{RC}(\mathcal{O}) = \text{FIN}$, *i.e.* $\text{RC}(\mathcal{O})$ *is the class of all finite languages.*

Proof. Let $\mathcal{O} \subseteq \{\cup, \cap\}$. The inclusion "$\supseteq$" is obvious, because any finite language $L = \{w_1, w_2, \ldots, w_n\}$ can be generated by an $\text{RC}(\mathcal{O})$-circuit with output nodes $\{v_1, v_2, \ldots, v_n\}$ such that the initial languages $\alpha(v_i) \stackrel{\text{df}}{=} \{w_i\}$ for $i = 1, 2, \ldots, n$.

"$\subseteq$" is also clear because since there is no concatenation, the largest language that can be generated is $\bigcup_{v \in V} \alpha(v)$ when V is the set of nodes and α is the function for the initial sets of the nodes. □

Corollary 5.2. $\text{RC}(\cup, \cap) \subsetneq \text{REG}$

Proof. Since $\text{RC}(\cup, \cap) = \text{FIN}$ by Theorem 5.1, this is obvious. □

The preceding theorem also implies that circuits without concatenation cannot really exploit their recurrence because an equivalent non-recurrent circuit can always be constructed (even a circuit without edges). On the other hand, a circuit that uses concatenation and whose nodes produce a new word in their languages in every step can also easily be constructed. We have already mentioned that the possibility of recurrence enables a circuit to generate non-regular languages. We will later show that $\text{REG} \subsetneq \text{RC}(\cdot)$, which means that recurrent circuits with concatenation can really make use of their recurrence and that there is a language $L \in \text{RC}(\cdot)$ such that no recurrent circuit generates L in a finite amount of steps.

5.2 Language Equations

We have already come up with the assertion that recurrent circuits correspond to language equations. In this section, we will formally define language equations and prove the mentioned equivalence, following the definitions and proofs by Okhotin. Systems of language equations are similar to normal systems of algebraic equations, only that instead of $+$ and $\cdot$, the operations $\cup, \cap$ and $\cdot$ are used and variables represent languages and not numbers. Additionally, these systems are always in a form such that on the left hand side of an equation there is always a single variable. We are looking for solutions to such equations, and especially for minimal solutions.

Definition (Regular expression with variables)**.**
Regular expressions over the alphabet Σ with variables $X_1, X_2, \ldots, X_n$ *are defined inductively in the following way:*

- ε*,* a *and* X_i *for* $a \in \Sigma$ *and* $i \in \{1, 2, \ldots, n\}$ *are regular expressions.*
- *If* φ_1, φ_2 *are regular expressions and* $\circ \in \{\cup, \cap, \cdot\}$*, then* $(\varphi_1 \circ \varphi_2)$ *is a regular expression.*

For the sake of brevity, we will omit the alphabet or the variables, if they are clear from the context.

We will not be so strict when we use these regular expressions, more specifically, we will leave out parentheses and make use of the associativity of the operations if it does not lead to confusion. Here, concatenation has the highest precedence followed by intersection and union in this order.

Such a regular expression can be seen as the definition of a function with variables $X_1, X_2, \ldots, X_n$. Next, we will define what it means if one substitutes the variables by values.

Definition (Evaluation of a regular expression with variables)**.**
Let φ *be a regular expression over the alphabet* Σ *with variables* $X_1, X_2, \ldots, X_n$*. The* evaluation mapping $\varphi\colon (\mathcal{P}(\Sigma^*))^n \to \mathcal{P}(\Sigma^*), L = (L_1, L_2, \ldots, L_n) \mapsto \varphi(L)$ *is defined inductively on the structure of* φ*:*

- $\varepsilon(L) \stackrel{df}{=} \{\varepsilon\}$
- $a(L) \stackrel{df}{=} \{a\}$ *for any* $a \in \Sigma$
- $X_i(L) \stackrel{df}{=} L_i$ *for any* $i \in \{1, 2, \ldots, n\}$
- *For regular expressions* φ_1, φ_2 *and* $\circ \in \{\cup, \cap, \cdot\}$*:*
  $(\varphi_1 \circ \varphi_2)(L) \stackrel{df}{=} \varphi_1(L) \circ \varphi_2(L)$

Definition (System of language equations)**.**
For $n \geq 1$, the tuple $\varphi = (\varphi_1, \varphi_2, \ldots, \varphi_n)$ is a system of language equations with n variables over the alphabet Σ *if for all $i \in \{1, 2, \ldots, n\}$, φ_i is a regular expression over the alphabet Σ with variables $X_1, X_2, \ldots, X_n$.*

The evaluation mapping φ is defined accordingly:
$\varphi\colon (\mathcal{P}(\Sigma^*))^n \to (\mathcal{P}(\Sigma^*))^n, L = (L_1, L_2, \ldots, L_n) \mapsto (\varphi_1(L), \varphi_2(L), \ldots, \varphi_n(L))$.

Definition (Solution of a system of language equations)**.**
A vector $L = (L_1, L_2, \ldots, L_n)$ is a solution of a system of language equations *φ if $L = \varphi(L)$.*

Of course, just from the definition, it is not clear if such a system always has a solution. And if it has a solution, it can also have more than one. We will see that these systems (i.e. allowing the operations $\cup, \cap$ and $\cdot$) always have a solution and that we get the least solution as the least fixed point of the evaluation function of a system of language equations.

Theorem 5.3. *[Okh02]*
$L = \bigcup_{i \in \mathbb{N}} \varphi^i(\emptyset^n)$ is the least solution of a system of language equations φ with n variables (moreover, such a solution always exists).
Here, least solution *is meant with respect to the partial order $\leq$ on equally-sized tuples of languages defined by $(L_1, L_2, \ldots, L_n) \leq (M_1, M_2, \ldots, M_n) \overset{df}{\iff} \bigwedge_{i=1}^{n} L_i \subseteq M_i$.*

We want to give examples for systems of language equations here. Let us start with a system that consists of only one equation:

Example 5.4. *$\varphi \overset{df}{=} (\varphi_1)$ with $\varphi_1 \overset{df}{=} a \cdot X_1 \cdot b \cup \varepsilon$.*
We now apply the evaluation mapping:

$$\begin{aligned}
\varphi_1(\emptyset) &= \{\varepsilon\} \\
\varphi_1^2(\emptyset) &= \{a\} \cdot \{\varepsilon\} \cdot \{b\} \cup \{\varepsilon\} = \{ab, \varepsilon\} \\
\varphi_1^3(\emptyset) &= \{a\} \cdot \{ab, \varepsilon\} \cdot \{b\} \cup \{\varepsilon\} = \{aabb, ab, \varepsilon\} \\
&\vdots
\end{aligned}$$

Obviously, $\bigcup_{i \in \mathbb{N}} \varphi_1^n(\emptyset) = \{a^n b^n \mid n \in \mathbb{N}\}$ and thus, by Theorem 5.3, the least solution of φ is $(\{a^n b^n \mid n \in \mathbb{N}\})$. We have to believe that it is the least solution,

but we can at least verify that it is a solution of the system by evaluating the expression:

$$\begin{aligned}\varphi_1(\{a^nb^n \mid n \in \mathbb{N}\}) &= \{a\} \cdot \{a^nb^n \mid n \in \mathbb{N}\} \cdot \{b\} \cup \{\varepsilon\} \\ &= \{a^{n+1}b^{n+1} \mid n \in \mathbb{N}\} \cup \{\varepsilon\} \\ &= \{a^nb^n \mid n \in \mathbb{N}\}\end{aligned}$$

We have already seen in Example 2.1 that this language is non-regular and context-free. The next example will present an even more complicated language that can be expressed by a system of language equations.

Example 5.5. $\varphi \stackrel{df}{=} (\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5)$*, where*

$$\begin{aligned}\varphi_1 &\stackrel{df}{=} X_2 \cdot X_3 \cap X_4 \cdot X_5 \\ \varphi_2 &\stackrel{df}{=} a \cdot X_2 \cdot b \cup \varepsilon & \varphi_4 &\stackrel{df}{=} a \cdot X_4 \cup \varepsilon \\ \varphi_3 &\stackrel{df}{=} c \cdot X_3 \cup \varepsilon & \varphi_5 &\stackrel{df}{=} b \cdot X_5 \cdot c \cup \varepsilon\end{aligned}$$

In this example, the equations $\varphi_2, \varphi_3, \varphi_4$ and φ_5 can be seen as independent systems of equations (after renaming the variables). It is obvious that the least solutions to these systems are also elements of the least solution to φ.

As we have seen in Example 5.4, the least solution for (φ_2) is $(\{a^nb^n \mid n \in \mathbb{N}\})$. Similarly, we get $(\{c\}^*)$ as least solution for (φ_3). The least solutions to (φ_4) and (φ_5) then are $(\{a\}^*)$ and $(\{b^nc^n \mid n \in \mathbb{N}\})$, respectively. Thus, we get as first element of the least solution for φ:

$$\{a^nb^n \mid n \in \mathbb{N}\} \cdot \{c\}^* \cap \{a\}^* \cdot \{b^nc^n \mid n \in \mathbb{N}\} = \{a^nb^nc^n \mid n \in \mathbb{N}\}$$

In Example 2.2, we have already noted that this language is context-sensitive but not context-free (although it is the intersection of two context-free languages) and thus an example of the expressive power of language equations with intersection, but we will get back to this later.

In the next step, we will define for each recurrent circuit one system of language equations such that each element of the system corresponds to exactly one gate of the circuit and both "generate" the same language.

Definition (Corresponding system of language equations)**.**
Let $\mathcal{O} \subseteq \{\cup, \cap, \cdot\}$. For any $\mathcal{O}$-circuit $C = (\Sigma, \{v_1, v_2, \ldots, v_n\}, V', E, \omega, \alpha)$,

the corresponding system of language equations φ_C *is a system of language equations with* n *variables over the alphabet* Σ*, defined as* $\varphi_C \stackrel{df}{=} (\varphi_1, \varphi_2, \ldots, \varphi_n)$ *such that for all* $i \in \{1, 2, \ldots, n\}$*:*
If $\text{pred}(v_i) = (v_{j_1}, v_{j_2}, \ldots, v_{j_k})$ *for* $k \geq 1$*:*

$$\varphi_i \stackrel{df}{=} \alpha(v_i) \cup \begin{cases} (X_{j_1} \cup X_{j_2} \cup \cdots \cup X_{j_k}) & \text{if } \omega(v_i) = \cup \\ (X_{j_1} \cap X_{j_2} \cap \cdots \cap X_{j_k}) & \text{if } \omega(v_i) = \cap \\ (X_{j_1} \cdot X_{j_2} \cdot \ldots \cdot X_{j_k}) & \text{if } \omega(v_i) = \cdot \end{cases}$$

and $\varphi_i \stackrel{df}{=} \alpha(v_i)$ *if* $|pred(v_i)| = 0$*.*

Theorem 5.6. *For any set of operations* $\mathcal{O} \subseteq \{\cup, \cap, \cdot\}$ *and any* $\mathcal{O}$*-circuit* $C = \{\Sigma, \{v_1, v_2, \ldots, v_n\}, V', E, \omega, \alpha\}$*, the least solution for the system of language equations* φ_C *is* $(C(v_1), C(v_2), \ldots, C(v_n))$*.*

Proof. By Lemma 4.1, we know that $(C(v_1), C(v_2), \ldots, C(v_n))$ is a solution for $\varphi_C = (\varphi_1, \varphi_2, \ldots, \varphi_n)$, so it remains to show that it is the least solution.

It suffices to argue that $C(v_i, t) \subseteq \varphi_i^{t+1}(\emptyset^n)$ for every $t \in \mathbb{N}$ and every $i \in \{1, 2, \ldots, n\}$ (here even equality holds). So let $i \in \{1, 2, \ldots, n\}$.

Induction basis. For every value of $\omega(v_i)$ we get $C(v_i, 0) = \alpha(v_i) \subseteq \varphi_i(\emptyset^n)$.

Induction step. Let $t \in \mathbb{N}$. If $|\text{pred}(v_i)| = 0$, the assertion is obvious, so let $\text{pred}(v_i) = (v_{j_1}, v_{j_2}, \ldots, v_{j_k})$ for $k \geq 1$.

$$\begin{aligned} &C(v_i, t+1) = \\ &\quad = C(v_i, t) \cup \begin{cases} (C(v_{j_1}, t) \cup C(v_{j_2}) \cup \cdots \cup C(v_{j_k})) & \text{for } \omega(v_i) = \cup \\ (C(v_{j_1}, t) \cap C(v_{j_2}) \cap \cdots \cap C(v_{j_k})) & \text{for } \omega(v_i) = \cap \\ (C(v_{j_1}, t) \cdot C(v_{j_2}, t) \cdot \ldots \cdot C(v_{j_k}, t)) & \text{for } \omega(v_i) = \cdot \end{cases} \\ &\quad \stackrel{\text{IH}}{\subseteq} \varphi_i^{t+1}(\emptyset^n) \cup \begin{cases} (\varphi_{j_1}^{t+1}(\emptyset^n) \cup \varphi_{j_2}^{t+1}(\emptyset^n) \cup \cdots \cup \varphi_{j_k}^{t+1}(\emptyset^n)) & \text{for } \omega(v_i) = \cup \\ (\varphi_{j_1}^{t+1}(\emptyset^n) \cap \varphi_{j_2}^{t+1}(\emptyset^n) \cap \cdots \cap \varphi_{j_k}^{t+1}(\emptyset^n)) & \text{for } \omega(v_i) = \cap \\ (\varphi_{j_1}^{t+1}(\emptyset^n) \cdot \varphi_{j_2}^{t+1}(\emptyset^n) \cdot \ldots \cdot \varphi_{j_k}^{t+1}(\emptyset^n)) & \text{for } \omega(v_i) = \cdot \end{cases} \\ &\quad = \varphi_i^{t+2}(\emptyset^n) \end{aligned}$$

□

The last step in the equivalence of systems of language equations and recurrent circuits is to show that every system of language equations can be "modeled" by a recurrent circuit. For that, we will use the following definition.

Definition (Modeled systems of language equations)**.**
A system of language equations $\varphi = (\varphi_1, \varphi_2, \ldots, \varphi_n)$ is modeled *by another system of language equations* $\psi = (\psi_1, \psi_2, \ldots, \psi_m)$*, for* $n, m \geq 1$ *with least solutions* $(L_1, L_2, \ldots, L_n)$ *and* $(M_1, M_2, \ldots, M_m)$*, respectively, if* $m \geq n$ *and* $L_i = M_i$ *holds for all* $1 \leq i \leq n$*.*

It is evident that for any system of language equations φ with n variables, there is a recurrent circuit C such that the corresponding system of language equations φ_C models φ. It is necessary that φ_C can have more variables than φ, because the expressions of φ can be arbitrarily nested, whereas a system of language equations that is the corresponding system for a recurrent circuit always follows the (limited) structure of the recurrent circuit.

5.3 RC($\cup, \cap, \cdot$) and Conjunctive Languages

In 2001, Okhotin introduced the notion of *conjunctive grammars* ([Okh01], an overview of current results can be found in [Okh03c]), which are an extension of the normal context-free grammars by a set-theoretic intersection. It will turn out that this extension is just the same as the extension of RC($\cup, \cdot$) to RC($\cup, \cap, \cdot$) and that the correspondence is very natural. The class of conjunctive languages CCFL, the languages generated by conjunctive grammars, is still contained in the context-sensitive languages and strictly contains even the intersection-closure of the context-free languages [Okh01].

We start with the definition of conjunctive grammars.

Definition (Conjunctive grammar)**.**
A conjunctive grammar $G = (\Sigma, N, N_0, P)$ *is defined as follows.*

1. Σ *is an alphabet (the* terminals*),*
2. N *is an alphabet disjoint from* Σ *(the* nonterminals*),*
3. $N_0 \in N$ *(the* start symbol*) and*
4. $P \subseteq N \times \{(\alpha_1, \alpha_2, \ldots, \alpha_n) \mid n \geq 1, \alpha_i \in (\Sigma \cup N)^* \text{ for } i = 1, 2, \ldots, n\}$ *is a finite set (the* productions*).*

In the following, we assume that the special symbol $\& \notin \Sigma \cup N$ and then also write $A \to \alpha_1 \& \alpha_2 \& \ldots \& \alpha_n$ for productions $(A, (\alpha_1, \alpha_2, \ldots, \alpha_n))$ for $n \geq 1$, $\alpha_i \in (\Sigma \cup N)^*$ and $i = 1, 2, \ldots, n$.

Next, we will define the languages generated by conjunctive grammars and see that we get exactly the notion of generation by context-free grammars if we restrict the right side of productions to tuples of size one.

Definition (Languages generated by a conjunctive grammars)**.**
Let $G = (\Sigma, N, N_0, P)$ be a conjunctive grammar where the special symbols $\&, (,) \notin \Sigma \cup N$. We write as shorthand $\Delta \stackrel{df}{=} \Sigma \cup N \cup \{\&, (,)\}$. The relation $\Rightarrow_G$ of derivability in one step *is defined as follows.*

*For any strings $w_1, w_2 \in \Delta^*$, $A \in N$ and any rule $A \to \alpha_1 \& \alpha_2 \& \ldots \& \alpha_n \in P$, $w_1 A w_2 \Rightarrow_G w_1(\alpha_1 \& \alpha_2 \& \ldots \& \alpha_n) w_2$ (*production rule*).*

*For any strings $w_1, w_2 \in \Delta^*$, any terminal string $w \in \Sigma^*$ and any number $n \geq 1$, $w_1(\underbrace{w \& w \& \ldots \& w}_{n}) w_2 \Rightarrow_G w_1 w w_2$ (*reduction rule*).*

The relation of derivability*, denoted as $\Rightarrow_G^*$, is defined as the reflexive-transitive closure of $\Rightarrow_G$.*

The language generated by *$A \in N$ is $G(A) \stackrel{df}{=} \{w \in \Sigma^* \mid A \Rightarrow_G^* w\}$ and the* language generated by the grammar *G is $L(G) \stackrel{df}{=} G(N_0)$.*

Finally, CCFL $\stackrel{df}{=} \{L(G) \mid G$ *is a conjunctive grammar*$\}$.

We will now use the theory of language equations to show the equivalence of conjunctive grammars and $\{\cup, \cap, \cdot\}$-circuits. Note that the following can be seen as a re-definition of a term, but we will see that it actually turns out to be the same. We will also see that the order in which the rules appear in the equations is not relevant.

Definition (Corresponding system of language equations)**.**
For a conjunctive grammar $G \stackrel{df}{=} (\Sigma, \{X_1, X_2, \ldots, X_n\}, X_1, P)$, the corresponding system of language equations *φ_G with n variables over the alphabet Σ is defined as $\varphi_G \stackrel{df}{=} (\varphi_1, \varphi_2, \ldots, \varphi_n)$ where we define for $i \in \{1, 2, \ldots, n\}$:*

$$\begin{aligned} \varphi_i \stackrel{df}{=} & (\alpha_{11} \cap \alpha_{12} \cap \cdots \cap \alpha_{1n_1}) \cup \\ & (\alpha_{21} \cap \alpha_{22} \cap \cdots \cap \alpha_{2n_2}) \cup \\ & \vdots \\ & (\alpha_{m1} \cap \alpha_{m2} \cap \ldots \alpha_{mn_m}) \end{aligned}$$

if $X_i \to \alpha_{j1} \& \alpha_{j2} \& \ldots \& \alpha_{jn_j}$ for $j = 1, 2, \ldots, m$ are the rules in P with A_i on the left side. If there are no such rules in P, then $\varphi_i \stackrel{df}{=} X_i$.

The following theorem is the analogon of Theorem 5.6 for conjunctive grammars.

Theorem 5.7. *[Okh02]*
Let $G \stackrel{df}{=} (\Sigma, \{X_1, X_2, \ldots, X_n\}, X_1, P)$ *be a conjunctive grammar. Then the least solution of* φ_G *is* $(G(X_1), G(X_2), \ldots, G(X_n))$.

Again, it is obvious that for any system of language equations φ there is a conjunctive grammar G such that the corresponding system of language equations φ_G models φ.

Theorem 5.8. $\mathrm{RC}(\cup, \cap, \cdot) = \mathrm{CCFL}$

Proof. For any $L \in \mathrm{RC}(\cup, \cap, \cdot)$ there is a recurrent $\{\cup, \cap, \cdot\}$-circuit $C = (\Sigma, \{V_1, V_2, \ldots, V_n\}, V', E, \omega, \alpha)$ such that $L(C) = L$. We can assume that $V' = \{V_1\}$. By the remark preceding this theorem, there is a conjunctive grammar $G = (\Sigma, \{N_1, N_2, \ldots, N_m\}, N_1, P)$ that has a corresponding system of language equations φ_G which models φ_C. Let $(L_1, L_2, \ldots, L_m)$ be the least solution to φ_G.

By the Theorems 5.6 and 5.7 we obtain, $L = L_1 = L(G)$.

The other direction, the construction of a $\{\cup, \cap, \cdot\}$-circuit from a conjunctive grammar, works analogously. □

Because of these close structural correspondences between recurrent $\{\cup, \cap, \cdot\}$-circuits and conjunctive grammars, many results by Okhotin can be transferred to circuits and $\{\cup, \cap, \cdot\}$-languages. We will now cite some of these results.

Theorem 5.9. *[Okh03c, Okh01, Okh03d]*
- $\Gamma_\cap(\mathrm{CFL}) \subsetneq \mathrm{CCFL} \subseteq \mathrm{P} \cap \mathrm{CSL}$
- *There are logspace-many-one-complete languages for* P *in* CCFL.
- *The problem* $\{(G, w) \mid G$ *is a conjunctive grammar and* $w \in L(G)\}$ *(the membership problem for conjunctive grammars) is* P*-complete.*

Note the fact that $\Gamma_\cap(\mathrm{CFL}) \subsetneq \mathrm{CCFL}$ indicates that a circuit that can use a recursive intersection is more powerful than a circuit that can only intersect (non-recursively) finitely many contex-free languages. Furthermore, it is obvious that $\mathrm{CCFL} \neq \mathrm{P}$ because of closure properties. Since CCFL contains P-complete problems, the logspace-reduction closure of CCFL equals P. The

strictness of the inclusion CCFL $\subseteq$ CSL is still an open problem. Finally, note that the membership-problem for context-free grammars is also already P-complete [GHR95].

Theorem 5.10. *[Okh01, Okh03b, Okh03a]*

- CCFL *is closed under* $\cup, \cap, \cdot, ^*, h^{-1}, \cap$REG *and* ^{R}.
- CCFL *is not closed under homomorphisms.*
- *If* P $\neq$ NP *then* CCFL *is not closed under* ε*-free homomorphisms.*

The last line of the previous theorem states that if CCFL is closed under ε-free homomorphisms, then P = NP. As this is very unlikely (though not completely impossible), this closure is not assumed to hold true. Another closure property, namely the closure of CCFL under complementation, is a completely open problem.

Recently, it has been shown by Jeż [Jeż07] that, in contrast to the context-free grammars (where this is Parikh's Theorem, [ABB97, Theorem 2.6]), conjunctive grammars over unary alphabets can generate non-regular languages. We will later show that this result can also be transferred to $\{\cap, \cdot\}$-circuits.

5.4 RC($\cup, \cdot$) and Context-Free Languages

We will see that one can easily reformulate a context-free grammar as an equivalent $\{\cup, \cdot\}$-circuit and vice versa. Informally, context-free grammars are nothing else than $\{\cup, \cdot\}$-circuits. It is not surprising that RC($\cup, \cdot$) = CFL and we will again prove it via language equations. This section only contains that proof, because we can use all the results about context-free languages for RC($\cup, \cdot$) and the context-free languages have been studied so intensely that we cannot hope to find any new properties.

First, we show that the conjunctive grammars are not only a semantical but also a syntactical extension of context-free grammars.

Lemma 5.11. *For every context-free grammar* $G = (\Sigma, N, N_0, P)$ *and for every conjunctive grammar* $G' = (\Sigma, N, N_0, \{(N, (\alpha)) \mid N \to \alpha \in P\})$ *it holds that* $G(A) = G'(A)$ *for every* $A \in N$.

Proof. Let $w \in G(A)$ for some $A \in N$. Every step of this context-free derivation obviously corresponds to a valid application of the production-rule in G'.

Since after all these productions have been made in G', there is no &-symbol in the word and thus it can be reduced to w, which means that $w \in G'(A)$.

For the other direction, let $w \in G'(A)$ for some $A \in N$. It is obvious, that in the derivation of w, all reduction rules can be applied after all production rules. If we again apply a context-free derivation rule for each of the production rules, we get a correct context-free derivation and thus $w \in G(A)$. □

Theorem 5.12. $\mathrm{RC}(\cup, \cdot) = \mathrm{CFL}$.

Proof. For the first inclusion, let $L \in \mathrm{RC}(\cup, \cdot)$. Then there is a recurrent $\{\cup, \cdot\}$-circuit $C = (\Sigma, \{v_1, v_2, \ldots, v_n\}, V', E, \omega, \alpha)$ such that $L(C) = L$. We can assume that $V' = \{v_1\}$. Obviously, φ_C does not use the operation $\cap$. Then there is a conjunctive grammar $G' = (\Sigma, N, N_0, P)$ that has a corresponding system of language equations $\varphi_{G'}$ that models φ_G, which means that $L = C(v_1) = G'(N_0) = L(G')$. Again, $\varphi_{G'}$ does not make use of the operation $\cap$. This means that the productions P are in the form $A \to (\alpha)$ for $A \in N, \alpha \in (\Sigma \cup N)^*$ and we can use Lemma 5.11, which states that a context-free grammar exists which generates the language $G'(N_0) = L(G')$. All together, this means that $L \in \mathrm{CFL}$.

The other inclusion can be shown analogously. □

We have already mentioned some properties of the context-free languages in section 2.2. They of course also hold for $\mathrm{RC}(\cup, \cdot)$-languages.

6 Circuits without Union

6.1 RC(·)

The class RC(·) can be seen as a restriction of CFL, namely the class of languages generated by context-free grammars that are restricted in a way such that there can be only one "complex" rule for each nonterminal. One could think that therefore RC(·) is deterministic context-free. We will show that this is not the case, RC(·) and DCFL are incomparable. Nevertheless, there is a restriction of nondeterministic pushdown-automata we will call "weakly nondeterministic pushdown-automata" which corresponds to the class RC(·).

Definition (WNPDA)**.**
The tuple $M = (\Sigma, \Delta, \delta, S)$ *is a* weakly nondeterministic pushdown automaton *(WNPDA) if*

1. Σ *is an alphabet (the* input alphabet*),*
2. Δ *is an alphabet (the* stack alphabet*),*
3. δ *is a finite function* $\delta\colon (\Sigma \cup \{\varepsilon\}) \times \Delta \to \mathcal{P}(\Delta^*)$ *(the* transition function*), such that for all* $A \in \Delta$ *there is a string of stack symbols* $w \in \Delta^+$ *such that for all* $a \in \Sigma \cup \{\varepsilon\}$, $\delta(a, A) \subseteq \{\varepsilon, w\}$.
4. $S \subseteq \Delta$ *(the* initial stack symbols*).*

The restriction on the transition function mentioned in 3. means that the automaton can write on the stack only the empty word or a non-empty string that is fixed for each read stack symbol.
We say that δ *contains the rule* $av \to w$ or $av \to w \in \delta$ if $w \in \delta(a, v)$ for $a \in \Sigma \cup \{\varepsilon\}$, $v \in \Delta$ and $w \in \Delta^*$.

Definition (Languages accepted by WNPDAs)**.**
Let $M = (\Sigma, \Delta, \delta, S)$ *be a WNPDA.* $(v, x) \in \Sigma^* \times \Delta^*$ *is a* configuration *of* M*, where* v *is the remaining input word and* x *is the word in the stack. Here, the stack grows to the left whereas the input is read from left to right. The* configuration transition relation $\overset{t}{\underset{M}{\rightsquigarrow}}$ *for* $t \in \mathbb{N}$ *is defined inductively as follows, where* $u, v \in \Sigma^*$, $x, y \in \Delta^*$ *and* $s \in \Delta$*:*

$$(u, x) \overset{0}{\underset{M}{\rightsquigarrow}} (v, y) \iff^{df} (u, x) = (v, y),$$

$$(u, sx) \overset{1}{\underset{M}{\rightsquigarrow}} (v, y) \iff^{df} u = av \text{ and } y \in \delta(a, s)x \text{ for some } a \in \Sigma \cup \{\varepsilon\},$$

$$(u, x) \overset{t+1}{\underset{M}{\rightsquigarrow}} (v, y) \iff^{df} (u, x) \overset{t}{\underset{M}{\rightsquigarrow}} (w, z) \overset{1}{\underset{M}{\rightsquigarrow}} (v, y) \text{ for some } (w, z) \in \Sigma^* \times \Delta^*.$$

Furthermore, $(u,x) \underset{M}{\rightsquigarrow} (v,y) \overset{df}{\iff} (u,x) \overset{t}{\underset{M}{\rightsquigarrow}} (v,y)$ *for some* $t \in \mathbb{N}$. *Finally, the* language accepted *by* M *is defined as*

$$L(M) \overset{df}{=} \{w \in \Sigma^* \mid (w,A) \underset{M}{\rightsquigarrow} (\varepsilon,\varepsilon) \text{ for some } A \in S\}.$$

Note that WNPDAs are restrictions of nondeterministic PDAs with multiple initial stack symbols that accept on empty stack. We will now show that these automata correspond to $\{\cdot\}$-circuits.

Proposition 6.1. *For any* $L \in \mathrm{RC}(\cdot)$ *there is a WNPDA* M *and an* ε*-free homomorphism* φ *such that* $L = \varphi(L(M))$.

Proof. Let $C = (\Sigma, V, V', E, \omega, \alpha)$ be a recurrent $\{\cdot\}$-circuit. We first modify the circuit so that we do not need the homomorphism in the rest of the proof. Let $\Sigma' \overset{\text{df}}{=} \{w \mid |w| \geq 1, w \in \alpha(v) \text{ for some } v \in V\}$ be an alphabet and $\varphi\colon \Sigma'^* \to \Sigma^*$ be an ε-free homomorphism defined via $\varphi(w) \overset{\text{df}}{=} w$ for $w \in \Sigma'$. Let further $C' \overset{\text{df}}{=} (\Sigma', V, V', E, \omega, \alpha')$, where $\alpha'(v) \overset{\text{df}}{=} \{w \in \Sigma' \cup \{\varepsilon\} \mid w \in \alpha(v)\}$ for $v \in V$. Note that every initial set of C' now contains words of length at most one (in the alphabet Σ'). Then, by the proof of Theorem 4.8, we know that $\varphi(L(C')) = L(C)$, because $\varphi \circ \alpha' = \alpha$.

It now remains to show that for every $\{\cdot\}$-circuit $C = (\Sigma, V, V', E, \omega, \alpha)$ such that every initial set of C contains words of length at most one there is a WNPDA M such that $L(C) = L(M)$. By Lemma 4.3, we assume that every node in C has exactly two predecessors (the construction is consistent with the assumption about the initial sets).

Then we define the WNPDA $M \overset{\text{df}}{=} (\Sigma, V, \delta, V')$ where δ contains the following rules for each $v \in V$:

- $av \to \varepsilon$ if $\alpha(v) = \{a\}$, $a \in \Sigma \cup \{\varepsilon\}$ and
- $\varepsilon v \to uw$ for $\mathrm{pred}(v) = (u,w)$.

Note that δ fulfills the requirements for a WNPDA transition function.

We first show one inclusion of $L(M) = L(C)$.

Claim 6.1.1. *For all* $t \in \mathbb{N}$, $w \in \Sigma^*$ *and* $v \in \Delta$: $w \in C(v,t) \Rightarrow (w,v) \underset{M}{\rightsquigarrow} (\varepsilon,\varepsilon)$.

Proof of the claim. We show this by induction over t.
Induction basis. Clearly, $w \in C(v,0) \Rightarrow w \in \alpha(v) \Rightarrow wv \to \varepsilon$ is a rule of $M \Rightarrow (w,v) \overset{1}{\underset{M}{\rightsquigarrow}} (\varepsilon,\varepsilon)$.

Induction step.
Let $w \in C(v, t+1)$. If $w \in C(v,t)$, then the assertion obviously holds. Otherwise, let $\mathrm{pred}(v) = (u_1, u_2)$. Then there are $w_1, w_2 \in \Sigma^*$ such that $w = w_1 w_2$ and $w_1 \in C(u_1, t)$ and $w_2 \in C(u_2, t)$. Then, we get $(w_i, u_i) \underset{M}{\rightsquigarrow} (\varepsilon, \varepsilon)$ for $i = 1, 2$ by the induction hypothesis and thus also $(w_1 w_2, u_1 u_2) \underset{M}{\rightsquigarrow} (\varepsilon, \varepsilon)$. And since M has a rule $\varepsilon v \to u_1 u_2$, we get $(w, v) \underset{M}{\rightsquigarrow} (\varepsilon, \varepsilon)$. □

And now we prove the other inclusion.

Claim 6.1.2. *For all $t \geq 1$, $w \in \Sigma^*$ and $v \in \Delta$: $(w, v) \overset{t}{\underset{M}{\rightsquigarrow}} (\varepsilon, \varepsilon) \Rightarrow w \in C(v)$.*

Proof of the claim. Again, we show this by induction over t.
Induction basis. If $(w, v) \overset{1}{\underset{M}{\rightsquigarrow}} (\varepsilon, \varepsilon)$ then $w \in \alpha(v)$ which in turn means that $w \in C(v)$.

Induction step.
Let $(w, v) \overset{t}{\underset{M}{\rightsquigarrow}} (\varepsilon, \varepsilon)$ and $\mathrm{pred}(v) = (u_1, u_2)$. Since $t > 1$, the automaton cannot use a rule where the stack is deleted in the first step and thus we must have $(w, v) \overset{1}{\underset{M}{\rightsquigarrow}} (w, u_1 u_2) \overset{t-1}{\underset{M}{\rightsquigarrow}} (\varepsilon, \varepsilon)$. Then there are $w_1, w_2 \in \Sigma^*$ with $w = w_1 w_2$ and $t_1, t_2 \geq 1$, $t_1 + t_2 = t - 1$ such that $(w_i, u_i) \overset{t_i}{\underset{M}{\rightsquigarrow}} (\varepsilon, \varepsilon)$ for $i = 1, 2$. This in turn implies $w_i \in C(u_i)$ for $i = 1, 2$ and since u_1, u_2 are the predecessors of v we get $w = w_1 w_2 \in C(u_1) C(u_2) \subseteq C(v)$. □

By the two claims we now have $(w, v) \underset{M}{\rightsquigarrow} (\varepsilon, \varepsilon) \iff w \in C(v)$ for every $w \in \Sigma^*, v \in V$. So M accepts w from start symbol v if and only if $w \in C(v)$. Since the start symbols of M are exactly the output gates of C, we get $L(M) = L(C)$. □

Now we have to show how to create a recurrent circuit from a WNPDA such that the circuit generates the language the WNPDA accepts.

Proposition 6.2. *For any WNPDA M, $L(M) \in \mathrm{RC}(\cdot)$.*

Proof. Let $M = (\Sigma, \Delta, \delta, S)$ be a WNPDA. We first construct a variant of a recurrent circuit where the only difference in the definition is that the initial

set $\alpha(v)$ for a node v can be any subset of $\Sigma \cup \{\varepsilon\}$, and not only the empty set or a singleton subset. We use $\hat{a}$ to denote subsets of $\Sigma \cup \{\varepsilon\}$.

Let now $C \stackrel{\text{df}}{=} (\Sigma, V, V'E, \omega, \alpha)$ where

- $V \stackrel{\text{df}}{=} \Delta \cup \mathcal{P}(\Sigma \cup \{\varepsilon\})$,
- $V' \stackrel{\text{df}}{=} \{S\}$,
- $\omega(v) \stackrel{\text{df}}{=} \cdot$ for all $v \in V$,
- $\alpha(A) \stackrel{\text{df}}{=} \{a \in \Sigma \cup \{\varepsilon\} \mid aA \to \varepsilon \in \delta\}$ for $A \in \Delta$,
- $\alpha(A) \stackrel{\text{df}}{=} A$ for $A \in \mathcal{P}(\Sigma \cup \{\varepsilon\})$

and E is implicitly defined via pred in the following way:

- For $A \in \mathcal{P}(\Sigma \cup \{\varepsilon\})$, $\text{pred}(A) \stackrel{\text{df}}{=} ()$.
- If $A \in \Delta$ and there is no rule of the form $aA \to w$ for $a \in \Sigma \cup \{\varepsilon\}$ and $w \in \Delta^+$, then $\text{pred}(A) \stackrel{\text{df}}{=} ()$.
- If $A \in \Delta$ and there is a rule of the form $aA \to w = U_1U_2 \dots U_n$ for $w \in \Delta^+$, $a \in \Sigma \cup \{\varepsilon\}$ and $U_i \in \Delta$ for $i = 1, 2, \dots, n$, then $\text{pred}(A) \stackrel{\text{df}}{=} (\{a \in \Sigma \cup \{\varepsilon\} \mid aA \to U_1U_2 \dots U_n \in \delta\}, U_1, U_2, \dots, U_n)$.

We now show, again by arguing for the two inclusions separately, that $L(C) = L(M)$.

Claim 6.2.1. *For every $t \in \mathbb{N}, A \in \Delta$ and every $w \in C(A, t)$ it holds that $(w, A) \underset{M}{\rightsquigarrow} (\varepsilon, \varepsilon)$.*

Proof of the claim. Induction basis. Let $A \in \Delta$ and $w \in C(A, 0)$. Then we have $w \in \alpha(A)$ and thus $wA \to \varepsilon \in \delta$, which in turn means that $(w, A) \underset{M}{\rightsquigarrow} (\varepsilon, \varepsilon)$.

Induction step. Let $t \geq 0, A \in \Delta$ and $w \in C(A, t+1)$.
If $w \in C(A, t)$, we are done. Otherwise, A must have predecessors, so let $\text{pred}(A) = (\hat{a}, U_1, U_2, \dots, U_n)$. Then there must be $a \in \hat{a}$ and $w_1, w_2, \dots, w_n \in \Sigma^*$ such that $w = aw_1w_2 \dots w_n$ and $w_i \in C(U_i, t)$ for $i = 1, 2, \dots, n$. Then we get $(w_i, U_i) \underset{M}{\rightsquigarrow} (\varepsilon, \varepsilon)$ for $i = 1, 2, \dots, n$, since $U_i \in \Delta$, and then also $(aw_1w_2 \dots w_n, U_1U_2 \dots U_n) \underset{M}{\rightsquigarrow} (\varepsilon, \varepsilon)$ which implies $(w, A) \underset{M}{\rightsquigarrow} (\varepsilon, \varepsilon)$. □

Claim 6.2.2. *For every $t \geq 1$, every $A \in \Delta$ and every $w \in \Sigma^*$ such that $(w, A) \overset{t}{\underset{M}{\rightsquigarrow}} (\varepsilon, \varepsilon)$, we have $w \in C(A)$.*

Proof of the claim. Induction basis. Let $A \in \Delta$, $w \in \Sigma^*$ and $(w, A) \overset{1}{\underset{M}{\rightsquigarrow}} (\varepsilon, \varepsilon)$. Then $wA \to \varepsilon \in \delta$ and thus $w \in \alpha(A) \subseteq C(A)$.

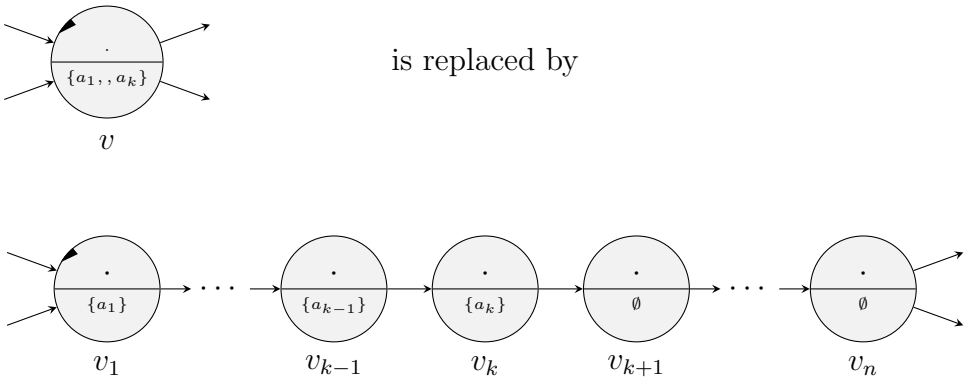


Figure 5: Replacement scheme used in the second part of the proof of Proposition 6.1.

Induction step. Let $A \in \Delta$, $w \in \Sigma^*$, $t \geq 1$ and $(w, A) \overset{t+1}{\underset{M}{\rightsquigarrow}} (\varepsilon, \varepsilon)$. Since $t+1 > 1$, M cannot start with the rule $wA \to \varepsilon$ but rather with one with non-empty right hand side like $aA \to U_1U_2 \dots U_n$ such that $a \in \Sigma \cup \{\varepsilon\}$, $n \geq 1$ and $U_i \in \Delta$ for $i = 1, 2, \dots, n$. If this rule exists, then A must have predecessors, so let $\mathrm{pred}(A) = (\hat{a}, U_1, U_2, \dots, U_n)$ such that $a \in \hat{a} \subseteq \Sigma \cup \{\varepsilon\}$ and $w = aw'$ for some $w' \in \Sigma^*$. Then we have $(w, A) \overset{1}{\underset{M}{\rightsquigarrow}} (w', U_1U_2 \dots U_n) \overset{t}{\underset{M}{\rightsquigarrow}} (\varepsilon, \varepsilon)$. Obviously, there must be $w_1, w_2, \dots, w_n \in \Sigma^*$ such that $w' = w_1w_2 \dots w_n$ and $(w_i, U_i) \overset{t_i}{\underset{M}{\rightsquigarrow}} (\varepsilon, \varepsilon)$ for $i = 1, 2, \dots, n$ and $\sum_{i=1}^{n} t_i = t$. This in turn means that $w_i \in C(U_i)$ for $i = 1, 2, \dots, n$ and thus $aw_1w_2 \dots w_n = w \in C(A)$. □

We can again say that the initial stack symbols are exactly the output gates and thus $L(M) = L(C)$. Note that this circuit C is not a real recurrent circuit, its initial languages can be bigger than allowed (any subset of $\Sigma \cup \{\varepsilon\}$). Thus it remains to show that these types of circuits can be simulated by "normal" recurrent circuits.

To this end, let $v \in V$ and let $n \overset{\text{df}}{=} |\Sigma \cup \{\varepsilon\}|$. We construct a circuit $C_2 \overset{\text{df}}{=} (\Sigma, V_2, E_2, V_2', \omega_2, \nu_2, \alpha_2)$ by replacing every node $v \in V$ by a chain of n new nodes according to the scheme in Figure 5.

Let the new nodes for v be $v_1, \dots, v_n \in V_2$. If $\alpha(v) = \{a_1, a_2, \dots, a_k\} \subseteq \Sigma \cup \{\varepsilon\}$, we define $\alpha_2(v_i) \overset{\text{df}}{=} \begin{cases} \{a_i\} & \text{for } 1 \leq i \leq k \\ \emptyset & \text{for } k < i \leq n \end{cases}$. Observe that C_2 is a correct recurrent circuit. Then, for every $t \in \mathbb{N}$ we get the relation $\alpha(v) \subseteq$

$C_2(v_1, t) \cup \{a_2, a_3, \ldots, a_k\} = C_2(v_2, t+1) \cup \{a_3, \ldots, a_k\} = \cdots = C_2(v_n, t+n-1)$, and thes especially $C_2(v_n, n-1) = C_1(v, 0) = \alpha(v)$.

We finally show that $L(C) = L(C_2)$ using the following claim.

Claim 6.2.3. *For every $t \in \mathbb{N}$ and $v \in V$, which is replaced by $v_1, v_2, \ldots, v_n$ in C_2, the following holds: $C(v, t) = C_2(v_n, nt + n - 1)$*

Proof of the claim. Induction basis. Let $v \in V$ be replaced by $v_1, v_2, \ldots, v_n$ in C_2. Then $C(v, 0) = \alpha(v) = C_2(v_n, n-1)$, as was show before.

Induction step. Let $t \in \mathbb{N}$, $w \in \Sigma^*$ and $v \in V$ be replaced by $v_1, v_2, \ldots, v_n$ in C_2. Let further $\mathrm{pred}(v) = (u_1, u_2, \ldots, u_k)$ and $\mathrm{pred}(v_1) = (u'_1, u'_2, \ldots, u'_k)$.

For the induction step we can assume that $w \notin C(v, t)$ and conclude

$$
\begin{aligned}
& w \in C(v, t+1) \\
\iff\ & w = w_1 w_2 \ldots w_k \text{ and } w_i \in C(u_i, t) \text{ for } i = 1, 2, \ldots, k \\
\overset{\text{IH}}{\iff}\ & w = w_1 w_2 \ldots w_k \text{ and } w_i \in C_2(u'_i, nt + n - 1) \text{ for } i = 1, 2, \ldots, k \\
\iff\ & w \in C_2(v_1, nt + n) \\
\iff\ & w \in C_2(v_n, nt + n + n - 1) = C_2(v_n, n(t+1) + n - 1)
\end{aligned}
$$

□

So obviously, $C(v) = C_2(v_n)$ if $v \in V$ was replaced by $v_1, v_2, \ldots, v_n$ in C_2, which means that $L(C) = L(C_2)$ and thus $L(M) = L(C_2) \in \mathrm{RC}(\cdot)$. □

Theorem 6.3. *$L \in \mathrm{RC}(\cdot)$ if and only if there is a WNPDA M and an ε-free homomorphisms φ such that $L = \varphi(L(M))$.*

Proof. If $L \in \mathrm{RC}(\cdot)$ the right hand side holds by Proposition 6.1. For the other direction, by Proposition 6.2, $L(M) \in \mathrm{RC}(\cdot)$ for any WNPDA M. Since $\mathrm{RC}(\cdot)$ is closed under homomorphisms by Theorem 4.8, $\varphi(L(M)) \in \mathrm{RC}(\cdot)$ also for any ε-free homomorphism. □

In the previous theorem, "ε-free homomorphism" can also be replaced by "homomorphism", since $\mathrm{RC}(\cdot)$ is closed under arbitrary homomorphisms by Theorem 4.8.

We now give some examples of languages that can be generated by $\{\cdot\}$-circuits.

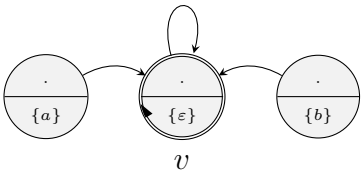


Figure 6: $\{\cdot\}$-circuit that generates $\{a^n b^n \mid n \in \mathbb{N}\}$.

Example 6.4. *The recurrent $\{\cdot\}$-circuit C shown in Figure 6 generates the language $L \stackrel{df}{=} \{a^n b^n \mid n \in \mathbb{N}\}$ from Example 2.1, which is context-free but not regular. This can be easily seen, because*

$$\begin{aligned} C(v,0) &= \{\varepsilon\}, \\ C(v,1) &= \{\varepsilon\} \cup \{a\} \cdot \{\varepsilon\} \cdot \{b\} = \{\varepsilon, ab\}, \\ C(v,2) &= \{\varepsilon, ab\} \cup \{a\} \cdot \{\varepsilon, ab\} \cdot \{b\} = \{\varepsilon, ab, aabb\}, \\ &\vdots \end{aligned}$$

so $C(v,t) = \{a^n b^n \mid 0 \leq n \leq t\}$ and thus $L(C) = C(v) = L$.

Next, we will show that the so-called *one-sided Dyck languages* are in RC(·). This is quite important, since in some sense they are generators for the whole class of context-free languages (cf. the Theorem of Chomsky and Schützenberger, Theorem 6.15). The one-sided Dyck languages are also called the *languages of matched or balanced parentheses*.

Definition (Dyck languages, D_n and $\mathcal{D}$)**.**
For all $n \geq 1$, we define the (one-sided) Dyck language over the alphabet $\Sigma_n \stackrel{\text{df}}{=} \bigcup_{i=1}^{n} \{l_i, r_i\}$ *or the* n-th Dyck language *as $D_n \stackrel{df}{=} L(G_n)$, where*
$G_n \stackrel{df}{=} (\Sigma_n, \{S\}, S, \{S \to \varepsilon, S \to SS, S \to l_1 S r_1, S \to l_2 S r_2, \ldots, S \to l_n S r_n\})$.

Furthermore, the class of all Dyck languages *is defined as $\mathcal{D} \stackrel{df}{=} \{D_n \mid n \geq 1\}$.*

So D_1 is the language of balanced parentheses with only one type of parentheses and D_n has exactly n types of parentheses.

Example 6.5. *The $\{\cdot\}$-circuit C_n shown in Figure 7 generates D_n for $n \geq 1$. To see this, we can translate this circuit into a system of language equations*

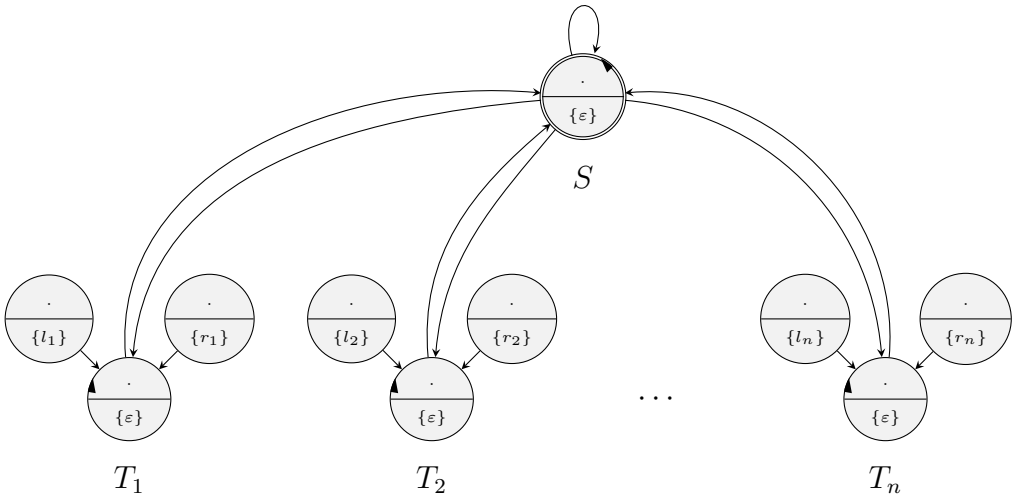


Figure 7: Circuit C_n that generates the n-th Dyck language D_n.

and that again into a context-free grammar. This grammar would have the following productions:

$$\begin{aligned} &S \to \varepsilon, && S \to ST_1T_2 \dots T_n, && \\ &T_i \to \varepsilon, && T_i \to l_i S r_i && \text{for } i = 1, 2, \dots, n \end{aligned}$$

These are equivalent to the next set of productions.

$$S \to \varepsilon, \qquad S \to S\overbrace{S \dots S}^{n}, \qquad S \to l_i S r_i \qquad \text{for } i = 1, \dots, n$$

And these in turn are equivalent to the following productions, which are exactly the productions of the grammar that generates the n-th Dyck language.

$$S \to \varepsilon, \qquad S \to SS, \qquad S \to l_i S r_i \qquad \text{for } i = 1, \dots, n$$

We now use the first example to show the non-closure of RC(·) under intersection. We will see later that the regular languages are included in RC(·) and (using the Dyck languages and the already mentioned Theorem of Chomsky and Schützenberger) show that RC(·) is not even closed under intersection with regular languages. That would imply the non-closure under intersection, but since the second proof is in a way non-constructive, we want to give the first proof nevertheless.

Theorem 6.6. RC(·) *is neither closed under intersection nor under complementation.*

Proof. Assume that RC(·) is closed under intersection. By Example 6.4, the language $\{a^n b^n \mid n \in \mathbb{N}\} \in \mathrm{RC}(\cdot)$. Since RC(·) is closed under concatenation and iteration by Proposition 4.6 and Proposition 4.7, the language from Example 2.2, $L \stackrel{\text{df}}{=} \{a^n b^n c^n \mid n \in \mathbb{N}\} = \{a^n b^n c^m \mid n, m \in \mathbb{N}\} \cap \{a^n b^m c^m \mid n, m \in \mathbb{N}\} = \{a^n b^n \mid n \in \mathbb{N}\} \cdot \{b\}^* \cap \{a\}^* \cdot \{b^m c^m \mid m \in \mathbb{N}\}$ and thus $L \in \mathrm{RC}(\cdot)$. Since L is not context-free and $\mathrm{RC}(\cdot) \subseteq \mathrm{RC}(\cup, \cdot) = \mathrm{CFL}$, we get a contradiction.

Since RC(·) is closed under union, but not under intersection, it cannot be closed under complementation by De Morgan's law. □

Corollary 6.7. $\mathrm{RC}(\cdot) \subsetneq \Gamma_\cap(\mathrm{RC}(\cdot))$

Proof. This is obvious, since RC(·) is not closed under intersection by Theorem 6.6. □

The next lemma tells us more about the structure of RC(·)-languages, we will use it to show that some languages are not in RC(·).

Lemma 6.8. *Let $S = (\Sigma, V, V', E, \omega, \alpha)$ be a recurrent $\{\cdot\}$-circuit. For every $v \in V'$ one of the following holds:*

1. *$S(v)$ is finite.*
2. *There are infinite sets $A, B \subseteq \Sigma^*$ and a finite set $C \subseteq \Sigma^*$ such that $S(v) = A \cdot B \cup C$.*
3. *There are finite sets $A, B, C, D, E, F \subseteq \Sigma^*$ such that $S(v) = \bigcup_{k=0}^{\infty} AB^kCD^kE \cup F$.*

Proof. Let $S = (\Sigma, V, V', E, \omega, \alpha)$ be a $\{\cdot\}$-circuit such that any gate in S has at most two predecessors (Lemma 4.2) and let $v \in V'$. If $S(v)$ is finite, we have case one, so let $S(v)$ be infinite. Then because of Lemma 4.1 there is a longest sequence of pairwise disjoint nodes $w = v_1, v_2, \ldots, v_r = v$ for $r \geq 1$ such that for all $i = 2, \ldots, r$: $v_{i-1} \in \mathrm{pred}(v_i)$, $S(v_{i-1})$ is infinite and for all other $u \in \mathrm{pred}(v_i)$, $S(u)$ is finite. We can now distinguish two cases:

1. $\mathrm{pred}(w) = (w_1, w_2)$ and $S(w_1)$ and $S(w_2)$ are both infinite. Then there are finite sets A, B, C such that $S(v) = A \cdot S(w) \cdot B \cup C$, by Lemma 4.1. Furthermore, there is a finite set D such that $S(v) = A \cdot S(w_1) \cdot S(w_2) \cdot B \cup D$. So the sets $A \cdot S(w_1)$ and $S(w_2) \cdot B$ are the infinite sets from the assumption and D is the finite set.

2. $v \in \text{pred}(w)$, so the sequence is a loop. Since all predecessors of nodes in the sequence generate infinite sets only if they are themselves part of the sequence, there are finite sets $A, B, C, D, E, F'F$ such that $S(v) = A \cdot S(w) \cdot E \cup F' = A \left(\bigcup_{k\in\mathbb{N}} B^k C D^k\right) E \cup F$. □

Let us discuss the three cases in the previous lemma. The meaning of the first case is clear. The second case means that all but finitely many words (those in C) are made up of two parts, where the left part of the word (from the set A) is completely independent from the right part of the word (from the set B). The third case implies a very simple structure on $C(v)$, because all the sets A, B, C, D, E, F are finite. Also note that the second case is not a generalization of the third case, because the sets D or B from case three can be empty.

We have already noted that RC(·) is not deterministic context-free, although the class can be captured by a type of pushdown-automaton that allows only a very restricted kind of nondeterminism. Next, we will show that $\{\cdot\}$-circuits can generate languages that are not deterministic context-free.

Proposition 6.9. $\text{RC}(\cdot) \not\subseteq \text{DCFL}$

Proof. Example 6.4 demonstrates that $\{a^n b^n \mid n \in \mathbb{N}\} \in \text{RC}(\cdot)$. Because RC(·) is closed under concatenation and union (Propositions 4.6 and 4.5), we get $L \stackrel{\text{df}}{=} \{a^n b^n c^m \# \mid n, m \in \mathbb{N}\} \cup \{a^n b^m c^m \mid n, m \in \mathbb{N}\} \in \text{RC}(\cdot)$, and L is a classic example of a language outside of DCFL. □

Although RC(·) seems rather limited, it contains all the regular languages.

Theorem 6.10. $\text{REG} \subsetneq \text{RC}(\cdot)$

Proof. We first show the inclusion. Every language in REG can be represented by a regular expression, i.e. a (finite) expression over single symbols using the operations $\{\cup, \cdot, ^*\}$. In fact, one can even show equality:

$$\text{REG} = \Gamma_{\cup,\cdot,^*}(\{\{a\} \mid a \text{ is a symbol or } a = \varepsilon\})$$

Since obviously $\{\{a\} \mid a \text{ is a symbol or } a = \varepsilon\} \subseteq \text{RC}(\cdot)$ and RC(·) is closed under union (Proposition 4.5), concatenation (Proposition 4.6) and iteration (Proposition 4.7), the inclusion holds.

The inclusion is proper, since the non-regular language $\{a^n b^n \mid n \in \mathbb{N}\} \in \text{RC}(\cdot)$ by Example 6.4. □

We will show that PAL $\stackrel{\text{df}}{=} \{x \in \{a,b\}^* \mid x = x^R\} \notin \text{RC}(\cdot)$ (cf. Example 2.3). For that we have to define some further notions. For two words $x, y \in \Sigma^*$, we say that x is a *prefix* of y, in symbols $x \sqsubseteq y$, if $y \in x\Sigma^*$. Furthermore, the *language of prefixes* of a certain word is defined as the set $P(w) \stackrel{\text{df}}{=} \{x \mid x \sqsubseteq w\}$ for a word w. We also extend P to languages L by $P(L) \stackrel{\text{df}}{=} \{x \mid x \sqsubseteq w \text{ for some } w \in L\}$.

Proposition 6.11. PAL $\notin$ RC(·)

Proof. Let $\Sigma \stackrel{\text{df}}{=} \{a,b\}$ and assume to the contrary that $S = (\Sigma, V, V', E, \omega, \alpha)$ is a $\{\cdot\}$-circuit such that $L(S) = \text{PAL}$. We now argue that for any $g \in V'$, the set generated by g must be *sparse*, which means there is a polynomial p such that $|\{x \in S(g) \mid |x| \leq n\}| \leq p(n)$. As the finite union of sparse sets is still sparse and PAL is clearly not sparse (it contains exponentially many words of each length), we get a contradiction.

Let $g \in V'$. If $S(g)$ is finite, then it is sparse. Otherwise, by Lemma 6.8 we have two remaining cases:

1. There are infinite sets $A, B \subseteq \Sigma^*$ and a finite set $C \subseteq \Sigma^*$ such that $S(g) = A \cdot B \cup C \subseteq \text{PAL}$. Observe that for any $x \in \Sigma^*$ if $x \in A$ then $B^R \subseteq P(x\Sigma^*)$, because otherwise $xB \subseteq \text{PAL}$ would not be true. If there are words $x, y, z \in \Sigma^*$ such that $zax, zby \in A$, then we have $B^R \subseteq P(zax\Sigma^*) \cap P(zby\Sigma^*) = P(z)$, which contradicts the fact that B is infinite. The opposite of this assumption is that for two words $x, y \in A$, we always have $x \sqsubseteq y$ or $y \sqsubseteq x$, and this means $A = \{x_0, x_1, x_2, \dots\}$ for some $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq \dots$. Then, $B^R = \bigcap_{i \in \mathbb{N}} P(x_i\Sigma^*)$, so B must also have this form, in particular, there is at most one word of length $n \in \mathbb{N}$ in B and in A. So the number of words in $A \cdot B$ of length exactly n is at most $n+1$ and thus $S(g)$ is sparse.
2. There are finite sets $A, B, C, D, E, F \subseteq \Sigma^*$ such that $S(g) = \bigcup_{i=0}^{\infty} AB^kCD^kE \cup F$. If one of the sets A, B, C, D, E is empty, then $S(g)$ is sparse. Otherwise, we will show that $B \subseteq P(x)$ for some $x \in \{a,b\}^*$. If $B = \{\varepsilon\}$, this is obviously the case. Otherwise there are $b_0 \in B - \{\varepsilon\}$, $a \in A$, $c \in C$, $d \in D$ and $e \in E$. For any $b \in B$ and $n \in \mathbb{N}$, the word $abb_0^ncd^{n+1}e \in \text{PAL}$. Since n can get arbitrarily large, there is an $x \in \{a,b\}^*$ such that $b \sqsubseteq x$ for any $b \in B$ and thus $B \subseteq P(x)$. Analogously, we get $D \subseteq P(y)$ for some $y \in \{a,b\}^*$, which obviously implies that $S(g)$ is sparse. □

The two languages PAL and D_1 seem quite similar, but the difference which

determines that PAL $\notin$ RC($\cdot$) but $D_1 \in$ RC($\cdot$) is that every word in D_1 can be divided into two independent parts just as is necessary in Lemma 6.8, whereas the dependencies between the left and the right end of a PAL-word are just too strong.

Corollary 6.12. RC($\cdot$) $\subsetneq$ CFL

Proof. The inclusion is shown by RC($\cdot$) $\subseteq$ RC($\cup, \cdot$) = CFL (Theorem 5.12). It is well-known that PAL $\in$ CFL (cf. Example 2.3), but Proposition 6.11 states that PAL $\notin$ RC($\cdot$). □

Next, we will see that the deterministic context-free languages are not in RC($\cdot$), which means that the automaton model WNPDA we defined earlier cannot accept all the languages a DPDA accepts. This can be made plausible because if a WNPDA writes a non-empty string on the stack, this string can only depend on the top stack symbol. A DPDA on the other hand can write different strings on the stack depending on the top stack symbol as well as the input symbol read.

Proposition 6.13. DCFL $\not\subseteq$ RC($\cdot$)

Proof. If DCFL was a subset of RC($\cdot$), then the palindromes with separation marker PALS $\stackrel{\text{df}}{=} \{wcw^R \mid w \in \{a,b\}^*\} \in$ DCFL (cf. Example 2.5) could be generated by a $\{\cdot\}$-circuit. Since RC($\cdot$) is closed under homomorphisms (Theorem 4.8), this would imply that the normal palindromes PAL could also be generated by a $\{\cdot\}$-circuit, which contradicts Proposition 6.11. □

We reformulate the propositions about RC($\cdot$) and DCFL in a corollary.

Corollary 6.14. RC($\cdot$) $\cap$ DCFL $\subsetneq$ RC($\cdot$) *and* RC($\cdot$) $\cap$ DCFL $\subsetneq$ DCFL.

Proof. We obviously only have to show the two inequalities. So assume that RC($\cdot$) $\cap$ DCFL = RC($\cdot$), but then RC($\cdot$) $\subseteq$ DCFL, which contradicts Proposition 6.9. Similarly, we get a contradiction with Proposition 6.13 for the assumption that RC($\cdot$) $\cap$ DCFL = DCFL. □

Next, we will show the non-closure of RC($\cdot$) under intersection with regular languages, we mentioned earlier. We first cite the Theorem of Chomsky and Schützenberger, because we will also use it in other places.

Theorem 6.15. *(Chomsky-Schützenberger, cf. [ABB97, Theorem 4.2])* $\mathrm{CFL} = h(\mathcal{D} \wedge \mathrm{REG})$*, which means that a language is context-free if and only if it is a homomorphic image of the intersection of a Dyck language with a regular language.*

Theorem 6.16. RC(·) *is not closed under intersection with regular languages.*

Proof. We assume to the contrary that RC(·) is closed under intersection with regular languages. Then, since $\mathcal{D} \subseteq \mathrm{RC}(\cdot)$ (cf. Example 6.5) and RC(·) is closed under homomorphisms (Theorem 4.8), we would get $\mathrm{RC}(\cdot) \supseteq h(\mathcal{D} \wedge \mathrm{REG}) = \mathrm{CFL}$ by Theorem 6.15. This contradicts Corollary 6.12. □

Proposition 6.17. $\mathrm{REG} \subsetneq \mathrm{RC}(\cdot) \cap \mathrm{DCFL}$

Proof. The inclusion $\mathrm{REG} \subseteq \mathrm{RC}(\cdot)$ has been shown in Theorem 6.10, the other inclusion is well-known. The inequality is due to $D_2 \in \mathrm{RC}(\cdot)$ and $D_2 \in \mathrm{DCFL} - \mathrm{REG}$, which is also a classical result. □

The next theorem will be proven indirectly using the theory of abstract families of languages.

Theorem 6.18. RC(·) *is not closed under inverse homomorphisms.*

Proof. We first show this property for a subset of RC(·), namely its *ε-free restriction* defined as $\mathrm{RC}^+(\cdot) \stackrel{\mathrm{df}}{=} \{L \in \mathrm{RC}(\cdot) \mid \varepsilon \notin L\}$ using an indirect proof. So assume that $\mathrm{RC}^+(\cdot)$ is closed under inverse homomorphisms.

From the theory of abstract families of languages it is known that an ε-free family of languages is closed under intersection with regular languages if it is closed under concatenation, ε-free homomorphisms and inverse homomorphisms ([MS97, Theorem 3.6] and [GH69]). An *ε-free family of languages* is a set of languages that contains at least one non-empty language and no language that contains the empty word. Note that $\mathrm{RC}^+(\cdot)$ possesses this property.

Since RC(·) is closed under concatenation by Proposition 4.6 and because the concatenation of two ε-free languages is always ε-free, $\mathrm{RC}^+(\cdot)$ is also closed under concatenation. Similarly, the closure of $\mathrm{RC}^+(\cdot)$ under ε-free homomorphisms follows from the closure of RC(·) under homomorphisms, which was shown in Theorem 4.8.

Thus by the cited theorem, $\mathrm{RC}^+(\cdot)$ would be closed under intersection with regular languages. We now proceed similarly to the proof of Theorem 6.16.

Since $\mathrm{CFL} = h(\mathcal{D} \wedge \mathrm{REG})$ (Theorem 6.15), it follows immediately that for a special unused symbol ∇, $\mathrm{CFL} = h(\nabla \cdot \mathcal{D} \wedge \nabla \cdot \mathrm{REG})$, since the homomorphism can simply erase the symbol ∇. Since $\nabla \cdot \mathcal{D} \subseteq \mathrm{RC}^+(\cdot)$ (cf. Example 6.5), we get $\mathrm{CFL} \subseteq h(\mathrm{RC}^+(\cdot) \wedge \mathrm{REG}) \subseteq h(\mathrm{RC}^+(\cdot)) \subseteq \mathrm{RC}(\cdot)$, because of the closure of $\mathrm{RC}(\cdot)$ under homomorphisms (Theorem 4.8). This obviously contradicts the fact that $\mathrm{RC}(\cdot) \subsetneq \mathrm{CFL}$, which was shown in Corollary 6.12.

So $\mathrm{RC}^+(\cdot)$ cannot be closed under inverse homomorphisms. Therefore, there is a language $L \in \mathrm{RC}^+(\cdot)$ and a homomorphisms h such that $h^{-1}(L) \notin \mathrm{RC}^+(\cdot)$. Since $\varepsilon \notin h^{-1}(L)$, we even get $h^{-1}(L) \notin \mathrm{RC}(\cdot)$ and thus the non-closure of $\mathrm{RC}(\cdot)$ under inverse homomorphisms. □

6.2 $\mathrm{RC}(\cap, \cdot)$

The next class we will investigate, $\mathrm{RC}(\cap, \cdot)$, is a restriction of the conjunctive grammars which allows intersection, but only a very weak union. Our main open question here is if $\mathrm{RC}(\cap, \cdot)$ is really different from $\mathrm{RC}(\cap, \cup, \cdot) = \mathrm{CCFL}$. In the other direction, we also cannot say whether $\Gamma_\cap(\mathrm{RC}(\cdot)) \neq \mathrm{RC}(\cap, \cdot)$ is true. We first show that this class is not closed under homomorphisms.

Theorem 6.19. $\mathrm{RC}(\cap, \cdot)$ *is not closed under homomorphisms.*

Proof. $\mathrm{RC}(\cap, \cdot)$ is closed under intersection by Proposition 4.6. Assume to the contrary that it is also closed under homomorphisms.

By the Theorem of Chomsky and Schützenberger (Theorem 6.15), we know that $\mathrm{CFL} = h(\mathcal{D} \wedge \mathrm{REG})$. Since $\mathcal{D} \subseteq \mathrm{RC}(\cdot)$ (cf. Example 6.5) and $\mathrm{REG} \subseteq \mathrm{RC}(\cdot)$ (Theorem 6.10), we get $\mathrm{CFL} \subseteq \Gamma_{h,\cap}(\mathrm{RC}(\cdot)) \subseteq \mathrm{RC}(\cap, \cdot)$.

Furthermore, since $\mathrm{RE} = h(\mathrm{DCFL} \wedge \mathrm{DCFL})$ [GGH67, Theorem 3.1], we get $\mathrm{RE} \subseteq \Gamma_{h,\cap}(\mathrm{DCFL}) \subseteq \Gamma_{h,\cap}(\mathrm{RC}(\cap, \cdot)) = \mathrm{RC}(\cap, \cdot)$, which is impossible. □

Note that the previous proof also shows the non-closure of $\Gamma_\cap(\mathrm{RC}(\cdot))$ under homomorphisms and thus does not separate $\mathrm{RC}(\cap, \cdot)$ and $\Gamma_\cap(\mathrm{RC}(\cdot))$.

We already mentioned that in contrast to context-free languages, conjunctive languages over unary alphabets can be non-regular. This was show by Jeż

[Jeż07] and we will see that the grammar he used in his proof is actually a grammar that can already be modeled by an $\{\cap, \cdot\}$-circuit, which means that this result directly transfers to RC($\cap, \cdot$).

Theorem 6.20. *[Jeż07]*
We define the system of language equations $\varphi \overset{df}{=} (\varphi_1, \varphi_2, \varphi_3, \varphi_4)$ *over the unary alphabet* $\{a\}$*, where*

$$\begin{aligned}
\varphi_1 &\overset{df}{=} (X_2 \cdot X_2 \cap X_1 \cdot X_3) \cup \{a\},\\
\varphi_2 &\overset{df}{=} (X_4 \cdot X_2 \cap X_1 \cdot X_1) \cup \{aa\},\\
\varphi_3 &\overset{df}{=} (X_4 \cdot X_4 \cap X_1 \cdot X_2) \cup \{aaa\},\\
\varphi_4 &\overset{df}{=} (X_3 \cdot X_3 \cap X_1 \cdot X_2).
\end{aligned}$$

If $L = (L_1, L_2, L_3, L_4)$ *is the least solution to* φ *then* $L_1 \notin$ REG*.*

Jeż showed that $L_1 = \{a^{4^n} \mid n \in \mathbb{N}\}$, which is obviously not regular.

Theorem 6.21. *The class* RC($\cap, \cdot$) *contains non-regular unary languages.*

Proof. We want to model the system of language equations from Theorem 6.20 by an $\{\cap, \cdot\}$-circuit. To this end, we define an index set for the gates $I \overset{\text{df}}{=} \{1, 2, 3, 4, 11, 12, 13, 22, 33, 42, 44\}$. Let $C \overset{\text{df}}{=} \{\{a\}, V, \{v_1\}, E, \omega, \alpha\}$ such that

$$\begin{aligned}
V &\overset{\text{df}}{=} \{v_i \mid i \in I\},\\
\alpha(v_i) &\overset{\text{df}}{=} \begin{cases} \{a^i\} & \text{for } i \in \{1,2,3\} \\ \emptyset & \text{otherwise,} \end{cases}\\
\omega(v_i) &= \begin{cases} \cap & \text{for } i \in \{1,2,3,4\} \\ \cdot & \text{otherwise,} \end{cases}
\end{aligned}$$

$$\begin{aligned}
\text{pred}(v_1) &\overset{\text{df}}{=} (v_{22}, v_{13}) & \text{pred}(v_2) &\overset{\text{df}}{=} (v_{42}, v_{11})\\
\text{pred}(v_3) &\overset{\text{df}}{=} (v_{44}, v_{12}) & \text{pred}(v_4) &\overset{\text{df}}{=} (v_{33}, v_{12})\\
\text{pred}(v_{ij}) &\overset{\text{df}}{=} (v_i, v_j) \quad \text{if } v_{ij} \in V.
\end{aligned}$$

Note that we have defined the relevant information for E in terms of the function pred.

The system of language equations φ_C over the variables $(X_i \mid i \in I)$ then is $\varphi_C = (\varphi_i \mid i \in I)$, where

$$\begin{aligned}
\varphi_1 &= (X_{22} \cap X_{13}) \cup \{a\},\\
\varphi_2 &= (X_{42} \cap X_{11}) \cup \{aa\},\\
\varphi_3 &= (X_{44} \cap X_{12}) \cup \{aaa\},\\
\varphi_4 &= (X_{33} \cap X_{12}),\\
\varphi_{ij} &= X_i \cdot X_j \text{ for } ij \in \{11, 12, 13, 22, 33, 42, 44\}.
\end{aligned}$$

It is obvious that φ_C models the system of language equations of Theorem 6.20. This means that the first component of the least solution, $L(C)$, is a non-regular unary language. □

Corollary 6.22. $\mathrm{RC}(\cap, \cdot) \nsubseteq \mathrm{CFL}$

Proof. We assume to the contrary that $\mathrm{RC}(\cap, \cdot) \subseteq \mathrm{CFL}$. By Theorem 6.21, there is a unary language L such that $L \in \mathrm{RC}(\cap, \cdot) - \mathrm{REG}$. Together with the assumption, this would imply $L \in \mathrm{CFL} - \mathrm{REG}$, but by Parikh's Theorem ([ABB97, Theorem 2.6]), $L \in \mathrm{CFL} \iff L \in \mathrm{REG}$, which is a contradiction. □

One could assume that the previous theorem also suggests the inequality $\mathrm{RC}(\cap, \cdot) \neq \Gamma_\cap(\mathrm{RC}(\cdot))$, because in the unary world, $\mathrm{RC}(\cdot) = \mathrm{REG}$ and the regular languages are closed under intersection. However, this is a false conclusion, since there can possibly be a unary language $L \in \Gamma_\cap(\mathrm{RC}(\cdot))$ such that $L = L_1 \cap L_2$ for some *non-unary* languages $L_1, L_2 \in \mathrm{RC}(\cdot)$, which would not be covered by our argumentation. So this theorem again does not separate $\mathrm{RC}(\cap, \cdot)$ from $\Gamma_\cap(\mathrm{RC}(\cdot))$.

The next theorem suggests that $\mathrm{RC}(\cap, \cdot)$ is not closed under ε-free homomorphisms and under inverse homomorphisms at the same time, because this would have rather unlikely consequences. Here we use the theory of abstract families of languages again.

Definition (Abstract family of languages)**.**
A set of languages $\mathcal{C}$ is an AFL *(abstract family of languages) if it contains a nonempty language and is closed under εh, h^{-1}, $\cap$REG, $\cup$ and $^+$. We will summarize all these operations under the operation aggregation* AFL*.*

Another useful ingredient for the proof of the next theorem is the class Q of *quasi-realtime languages*, the class of languages accepted by nondeterministic multitape Turing-machines in realtime, first defined in [BG69]. There it was also shown that Q is the least AFL that contains the Dyck language D_2 and is closed under intersection, which means $\mathrm{Q} = \Gamma_{\mathrm{AFL},\cap}(\{D_2\})$.

Theorem 6.23. *If* $\mathrm{RC}(\cap, \cdot)$ *is closed under* ε*-free homomorphisms and under inverse homomorphisms, then* $\mathrm{P} = \mathrm{NP}$*.*

Proof. If $\mathrm{RC}(\cap, \cdot)$ is closed under ε-free homomorphisms and under inverse homomorphisms, then $\mathrm{RC}(\cap, \cdot)$ is obviously an AFL that is additionally closed under intersection (see Table 1). Since $D_2 \in \mathrm{RC}(\cdot) \subseteq \mathrm{RC}(\cap, \cdot)$, we get $\mathrm{Q} \subseteq \mathrm{RC}(\cap, \cdot)$.

By $\mathcal{R}^{\mathrm{p}}_{\mathrm{m}}(\mathcal{C})$ we denote the closure of the class $\mathcal{C}$ under polynomial time many-one reductions. It is easy to see that $\mathcal{R}^{\mathrm{p}}_{\mathrm{m}}(\mathrm{Q}) = \mathrm{NP}$, because we can use standard padding techniques to change *realtime* to *polynomial time*.

Then we have $\mathrm{NP} = \mathcal{R}^{\mathrm{p}}_{\mathrm{m}}(\mathrm{Q}) \subseteq \mathcal{R}^{\mathrm{p}}_{\mathrm{m}}(\mathrm{RC}(\cap, \cdot) \subseteq \mathcal{R}^{\mathrm{p}}_{\mathrm{m}}(\mathrm{P}) = \mathrm{P}$. Since obviously $\mathrm{P} \subseteq \mathrm{NP}$, we have equality. □

This also means that $\mathrm{RC}(\cap, \cdot)$ cannot be an AFL unless $\mathrm{P} = \mathrm{NP}$. The proof also works for $\mathrm{RC}(\cap, \cup, \cdot)$. There Okhotin already showed the closure under inverse homomorphisms [Okh03a] and remarked that the additional closure under ε-free homomorphism would imply $\mathrm{P} = \mathrm{NP}$ [Okh03b].

Finally, we want to mention an obvious closure property.

Proposition 6.24. $\mathrm{RC}(\cap, \cdot)$ *is closed under intersection with regular languages.*

Proof. $\mathrm{RC}(\cap, \cdot)$ is closed under intersection by Proposition 4.6. Furthermore, this class trivially includes $\mathrm{RC}(\cdot)$, which includes the regular languages by Theorem 6.10 and thus the assertion holds. □

We obviously have the inclusions $\Gamma_{\cap}(\mathrm{RC}(\cdot)) \subseteq \mathrm{RC}(\cap, \cdot) \subseteq \mathrm{RC}(\cup, \cap, \cdot) = \mathrm{CCFL}$, but since $\Gamma_{\cap}(\mathrm{RC}(\cdot)) \subseteq \Gamma_{\cap}(\mathrm{CFL}) \neq \mathrm{CCFL}$ at least one of these two inclusions must be a strict one, but which one of them is it? One can reformulate this question more figuratively: Where is a "recursive" intersection or a "recursive" union really more than a "non-recursive" one? (Remember

that every circuit-class is closed under finite (non-recursive) union.) Finding a method to prove languages to be non-conjunctive is an open question for conjunctive grammars, as one hopes to show some new results, for example CCFL $\subsetneq$ CSL, this way. Similarly, a method to prove languages to be not in RC($\cap, \cdot$) could separate RC($\cap, \cdot$) from CCFL. Unfortunately, we could not find such methods. Another way to prove strict inclusion of RC($\cap, \cdot$) in CCFL would be to show CFL $\not\subseteq$ RC($\cap, \cdot$). This non-inclusion seems reasonable, because RC($\cap, \cdot$) suffers from the same limitations concerning unions as RC($\cdot$) does and RC($\cdot$) is strictly included in the context-free languages.

7 Summary of Results and Open Problems

Operation	REG	RC($\cdot$)	RC($\cup, \cdot$) (CFL)	RC($\cap, \cdot$)	RC($\cup, \cap, \cdot$) (CCFL)
$\cup$	+ 2.6	+ 4.5	+ 2.7	+ 4.5	+ 5.10
$\cap$	+ 2.6	− 6.6	− 2.7	+ 4.6	+ 5.10
$^-$	+ 2.6	− 6.6	− 2.7		
$\cdot$	+ 2.6	+ 4.6	+ 2.7	+ 4.6	+ 5.10
*	+ 2.6	+ 4.7	+ 2.7	+ 4.7	+ 5.10
h	+ 2.6	+ 4.8	+ 2.7	− 6.19	− 5.10
εh	+ 2.6	+ 4.8	+ 2.7	cf. 6.23	$\Rightarrow$ P = NP 5.10
h^{-1}	+ 2.6	− 6.18	+ 2.7	cf. 6.23	+ 5.10
$\cap$REG	+ 2.6	− 6.16	+ 2.7	+ 6.24	+ 5.10
^{R}	+ 2.6	+ 4.9	+ 2.7	+ 4.9	+ 5.10

Table 1: Closure properties of most classes of recurrent circuits and related classes. "+" means that the respective closure holds, "−" means that it does not hold, while an empty cell indicates that we do not know whether the property holds or not. The numbers denote where the result was obtained. The entry "$\Rightarrow$ P = NP" means that this closure would imply P = NP and thus is very unlikely.

We have listed the closure properties of the recurrent-circuit-classes except for RC($\cup, \cap$) = FIN in Table 1. The entries "cf. 6.23" denote that if the class RC($\cap, \cdot$) is closed both under inverse and ε-free homomorphisms, then P = NP, as shown in Theorem 6.23. Note that none of the new classes RC($\cdot$) and RC($\cap, \cdot$) is an AFL, both lack the closure under inverse homomorphisms. This closure is also cruicial for classes being a *cone* (families of languages that are closed under h, h^{-1} and $\cap$REG), which is a more general abstract type of families.

In Figure 8, the classes discussed in this thesis are related to each other and to other important classes of formal languages and complexity theory in terms of inclusions. The inclusions hold upwards while dashed lines represent inclusions and solid ones represent proper inclusions. The numbers beneath the lines denote where the result is stated in. If a line is unlabeled, the inclusion is trivial or a classical result. The region marked as "unary $\subseteq$ REG" includes all the classes that are included in the regular languages when restricted to unary alphabets, whereas the region marked as "unary $\not\subseteq$ REG" includes all

the classes where it has been shown that they contain a non-regular unary language (cf. 6.21).

We have seen that recurrent circuits offer a new model to describe context-free languages. Furthermore, they can be used to define restrictions and extensions of context-free languages. The class $\mathrm{RC}(\cdot)$ is of particular interest as a new class of formal languages. It is properly included in the context-free languages and contains the regular languages. $\mathrm{RC}(\cdot)$ cannot be modeled by deterministic pushdown-automata but nevertheless, there are pushdown-automata that need only very limited nondeterminism and almost capture $\mathrm{RC}(\cdot)$.

The most powerful class of recurrent circuits is equal to a class that has been defined only recently, the conjunctive languages. They have some very nice properties, for example their grammars can be parsed efficiently while they are more powerful than simple context-free grammars. Important open questions for these languages are the closure under complementation, under ε-free homomorphisms and the proper inclusion in the context-sensitive languages. We could not find any new techniques to solve these problems using recurrent circuits.

The class $\mathrm{RC}(\cap, \cdot)$ is very similar to the general case and indeed, they share many closure properties. This class contains a non-regular unary language, as do the conjunctive languages (remember that this is not the case for the context-free languages). There are even some problems that are open for $\mathrm{RC}(\cap, \cdot)$ and for conjunctive languages. It seems that a method to prove non-trivial non-inclusions in $\mathrm{RC}(\cap, \cdot)$ would also help to prove this for conjunctive languages, which is still an important open question. Unsolved problems concerning the class $\mathrm{RC}(\cap, \cdot)$ are:

- Is $\mathrm{RC}(\cap, \cdot)$ closed under ε-free homomorphism (unconditional result)?
- Is $\mathrm{RC}(\cap, \cdot)$ closed under inverse homomorphism (unconditional result)?
- Is $\mathrm{RC}(\cap, \cdot)$ closed under complementation?
- What would be a method to prove that a language is not included in $\mathrm{RC}(\cap, \cdot)$?
- Which of the inclusions $\Gamma_\cap(\mathrm{RC}(\cdot)) \subseteq \mathrm{RC}(\cap, \cdot) \subseteq \mathrm{RC}(\cap, \cup, \cdot) = \mathrm{CCFL}$ are strict?

In the definition of recurrent circuits, we restricted the initial sets of each node to sets containing single words. Interesting alternative definitions would be to allow finite sets or also whole regular languages. This is reasonable because those are basic sets. This modification would not change $\mathrm{RC}(\cup, \cap, \cdot)$ and $\mathrm{RC}(\cup, \cdot)$, but it could change the other classes significantly, so that we

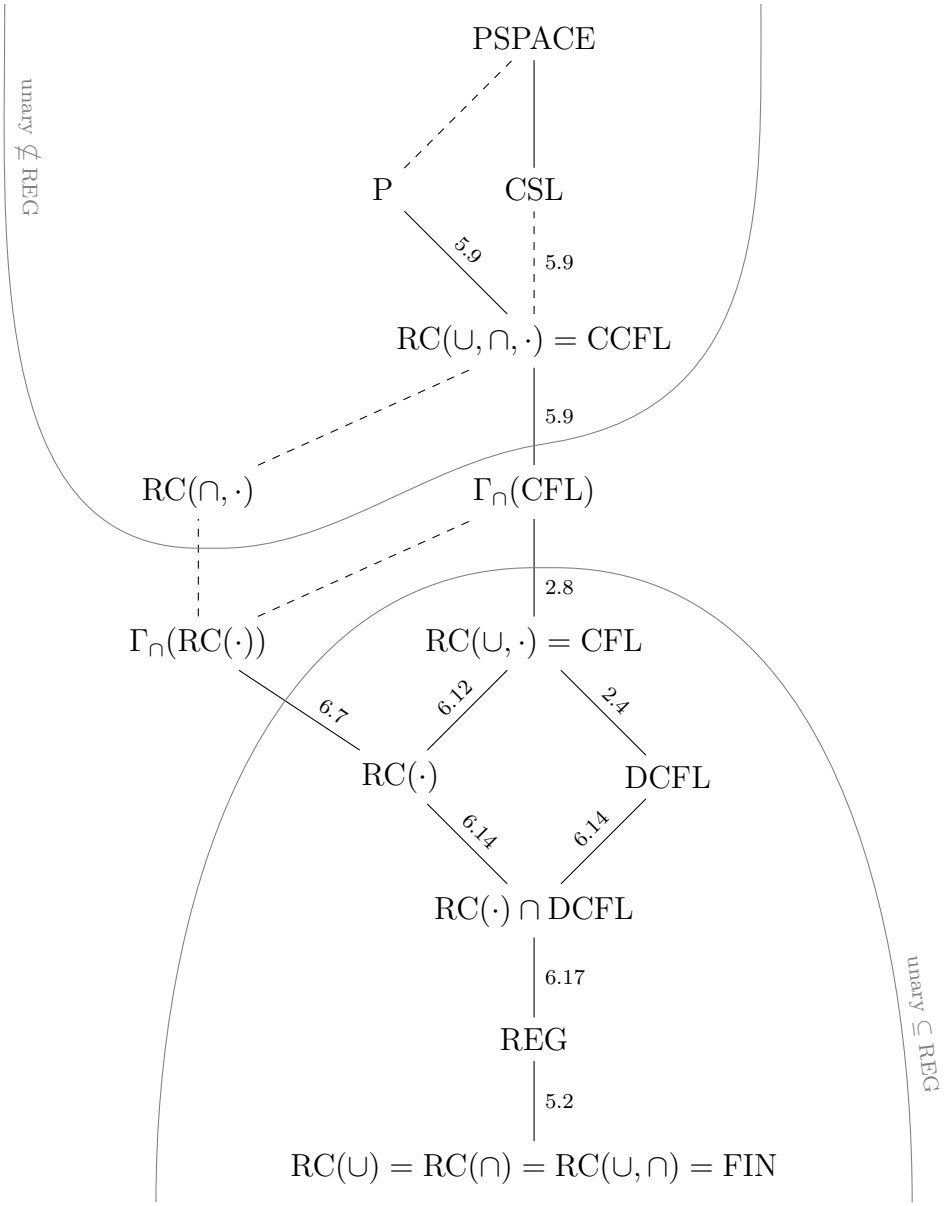


Figure 8: Inclusion diagram for the classes discussed in this thesis. See the text for an explanation of the notations.

could perhaps obtain classes that have better closure properties or fit exactly to some alternative characterization.

Another way to define concepts similar to these recurrent circuits would be to extend the set of available operations. The complementation would complete the boolean set-theoretic operations, but when one leaves the rest of the definition as it is, the natural correspondence to language equations would not persist anymore. Okhotin studied language equations with complementation and observed that they do not always have a least solution, whereas a recurrent circuit is forced to generate a unique language by the definition. As long as one stays with monotone operations, the correspondence to language equations stays intact.

Furthermore, one could consider iteration as its own operation. Of course, if concatenation is also available, this does not change anything, since we can already simulate the Kleene star with a single concatenation gate, but the other classes could be interesting.

Finally, the complexity of different kinds of problems connected with recurrent circuits can be considered. We have already mentioned that the membership problem for conjunctive grammars is P-complete (Theorem 5.9), which also transfers to recurrent $\{\cup, \cap, \cdot\}$-circuits. The membership problem for context-free grammars is also P-complete. Does this completeness still hold for $\{\cdot\}$-circuits? Other problems that could be of interest include emptiness and finiteness of the generated language, inclusion of generated languages, equivalence of recurrent circuits and others. For the equivalence problem note that this is undecidable for context-free grammars, but it is decidable for deterministic pushdown-automata, so this could be of particular interest for $\{\cdot\}$-circuits and WNDPAs.

References

[ABB97] J.-M. Autebert, J. Berstel, and L. Boasson. Context-free languages and pushdown automata. In *Handbook of Formal Languages, Vol. 1: Word, Language, Grammar*, pages 111–174. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[BG69] R. V. Book and S. A. Greibach. Quasi-realtime languages (extended abstract). In *STOC '69: Proceedings of the first annual ACM symposium on Theory of computing*, pages 15–18, New York, NY, USA, 1969. ACM Press.

[GGH67] S. Ginsburg, S. A. Greibach, and M. Harrison. One-way stack automata. *Journal of the ACM*, 14(2):389–418, 1967.

[GH69] S. Greibach and J. E. Hopcroft. Independence of AFL operations. *Memoirs of the AMS*, 87:33–40, 1969.

[GHR95] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press, 1995.

[HMU01] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 2nd edition, 2001.

[Jeż07] A. Jeż. Conjunctive grammars can generate non-unary regular languages. *Developments in Language Theory*, 2007. to appear.

[MS97] A. Mateescu and A. Salomaa. Aspects of classical language theory. In *Handbook of Formal Languages, Vol. 1: Word, Language, Grammar*, pages 175–251. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[Okh01] A. Okhotin. Conjunctive grammars. *Journal of Automata, Languages and Combinatorics*, 6(4):519–535, 2001.

[Okh02] A. Okhotin. Conjunctive grammars and systems of language equations. *Programming and Computer Software*, 28(5):243–249, 2002.

[Okh03a] A. Okhotin. Conjunctive languages are closed under inverse homomorphism. Technical Report 2003-468, School of Computing, Queen's University, Kingston, Ontario, Canada, 2003.

[Okh03b] A. Okhotin. The hardest linear conjunctive language. *Information Processing Letters*, 86(5):247–253, 2003.

[Okh03c] A. Okhotin. An overview of conjunctive grammars, Formal Language Theory Column. *Bulletin of the EATCS*, 79:145–163, 2003.

[Okh03d] A. Okhotin. A recognition and parsing algorithm for arbitrary conjunctive grammars. *Theoretical Computer Science*, 302(1-3):365–399, 2003.

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Würzburg, den 29. August 2007

.....
Christian Reitwießner