Deque languages, automata and planar graphs

Stefano Crespi Reghizzi and Pierluigi San Pietro

Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) and CNR - IEIIT
Politecnico di Milano, Piazza Leonardo da Vinci 32, Milano I-20133
stefano.crespireghizzi@polimi.it   pierluigi.sanpietro@polimi.it

Abstract. The memory of a *deque* automaton is more general than a queue or two stacks; to avoid overgeneralization, we consider quasi-real-time operation. Normal forms of such automata are given. Deque languages form an AFL but not a full one. We define the characteristic deque language, CDL, which combines Dyck and AntiDyck (or FIFO) languages, and homomorphically characterizes the deque languages. The notion of deque graph, from graph theory, well represents deque computation by means of a planar hamiltonian graph on a cylinder, with edges visualizing producer-consumer relations for deque symbols. We give equivalent definitions of CDL by labelled deque graphs, by cancellation rules, and by means of shuffle and intersection of simpler languages. The labeled deque graph of a sentence generalizes traditional syntax trees. The layout of deque computations on a cylinder is remindful of 3D models used in theoretical (bio)chemistry.

1   Introduction

This research pertains to the classical investigations on languages recognized by automata equipped with various types of auxiliary memory, such as pushdown stacks, queues, and combinations thereof. Introduced by D. Knuth [9], the *deque* data-type is common in computer science, where it is typically implemented by means of a bidirectional buffer. A deque memory combines the standard operations of a queue and of two stacks but, unlike such simpler cases which impose serialization, it permits parallel execution or interleaving of some operations. Work on multihead/multi-tape Turing machines has also studied simulation of deques, stacks and queues, e.g, in [11], where a deque is simulated in realtime by a machine with four single head tapes; see also [12] for deque simulation using stacks. On the other hand, the deque has been rarely studied as a formal language model: the only deque automaton model we know of is in [3], but it is restricted and contains errors corrected in [5].

First, the present study of deque automata is an attempt to fill such gap and to establish formal relations especially with the family of queue automata [15, 14, 6], which has recently attracted renewed interest (a survey is in [10]). A second reason to investigate deque languages is that they seem to fit, better than traditional automata, with the linguistic models proposed by molecular biology and chemistry, to study spatial arrangements of macromolecule sequences. Such fitting is suggested by a similarity of representation, next outlined. In their investigation of plane drawings of graphs on cylinder surfaces, Auer *et al*. [1, 2] showed that "a plane drawing is possible if, and only if, the graph is a . . . deque graph, i.e., the vertices of the graph can be processed according to a linear order and the edges correspond to items in the deque inserted and

removed at their end vertices". Similar more complex embeddings of planar graphs on a cylinder are considered in natural sciences, e.g., in [4, 13] for chemistry and for RNA.

Paper content and contributions. Sect. 2 defines deque automata (DA) consistently with existing definitions of pushdown and queue automata (PDA, QA). To prevent spontaneous moves to turn the deque without reading from input, thus simulating a Turing machine, we focus on quasi-real-time (QRT) operations as in QA [6]. Our DA definition is rather robust, as the corresponding language family, denoted as $\mathcal{F}(DA)$, is unaffected by some typical variations: number of states, number of deque symbols processed per move, and restriction to real-time computations. The family $\mathcal{F}(DA)$ forms an abstract family of languages (AFL) but not a full one. We define a convenient normal form using distinct tape alphabets for operations on stacks and on queues and performing at most one deque operation per move.

To illustrate expressivity, we introduce a family of languages featuring any number and ordering of reversed and directed replications. Sect. 2.1 defines the *characteristic deque language* (CDL) which plays the role of the Dyck and the AntiDyck (a.k.a. FIFO) language, resp. for CF and queue languages and yields a Chomsky-Schützenberger theorem for $\mathcal{F}(DA)$. Sect. 3 exploits the planar cylindrical graphs recently defined in [2] and develops them into a technique for analyzing and visualizing deque automata and languages. We prove that CDL computations on a DA are exactly represented by a *labeled deque graph* (LDG). Then we characterize CDL by cancellation rules, unifying the classical rules for Dyck and AntiDyck languages. A closed formula can express a CDL as the shuffle of two CF languages intersected with a queue language; its corollary for the whole $\mathcal{F}(DA)$ family immediately follows. We end by showing how to extend the deque graph representing CDL words to all DA languages, thus endowing this family with a sort of structured syntax visualization. The conclusion mentions directions for future research.

2   Basic definitions and properties

A double ended queue or *deque* is an unbounded tape containing a possibly empty string of symbols from an alphabet Γ. In a horizontal layout, the left end and the right end of the tape are resp. called *front* and *tail*, both equipped with a reading and writing head. The four deque *operation types* are: write/read a symbol at front, write/read a symbol at tail. Each reading operation cancels the symbol. After each writing (resp. reading) operation at front, the front head moves left (resp. right) by one case; the tail head resp. moves right/left after writing/reading. The deque is empty when the front and tail heads coincide.

A *deque automaton M* is a nondeterministic acceptor with one-way read-only input tape, finite-state control, and deque memory tape, which initially is empty. M is restricted to operate in QRT.

Definition 1. *A QRT* deque automaton *(DA) is a 6-tuple $M = (\Sigma, \Gamma, Q, q_0, \delta, F)$ where Σ is the terminal alphabet, Γ the finite deque alphabet, $q_0 \in Q$ and $F \subseteq Q$ resp. the initial and final states; and the transition relation δ is a finite subset:*

$$\delta \subseteq (Q \times (\Sigma \cup \varepsilon) \times \Gamma^* \times \Gamma^* \times Q \times \Gamma^* \times \Gamma^*)$$

For each 7-tuple, called a *transition*, $(q, a, \gamma_l, \gamma_r, q', \sigma, \rho) \in \delta$, q and q' are resp. the present and next state, γ_l, σ are the words resp. read and written on deque front, and γ_r, ρ are the words resp. read and written on deque tail. If $a = \varepsilon$ then the tuple is called an ε-transition.

A configuration is a 3-tuple $(q, u, \gamma) \in Q \times \Sigma^* \times \Gamma^*$. A move is a binary relation $\rightarrow_M$ on configurations defined as follows. We say that $(q, au, \gamma_l \gamma \gamma_r) \rightarrow_M (q', u, \sigma \gamma \rho)$ if $q, q' \in Q, a \in \Sigma \cup \varepsilon, u \in \Sigma^*, \sigma, \rho, \gamma, \gamma_l, \gamma_r \in \Gamma^*$ and the transition $(q, a, \gamma_l, \gamma_r, q', \sigma, \rho) \in \delta$. A move where $a = \varepsilon$ is called *spontaneous*.
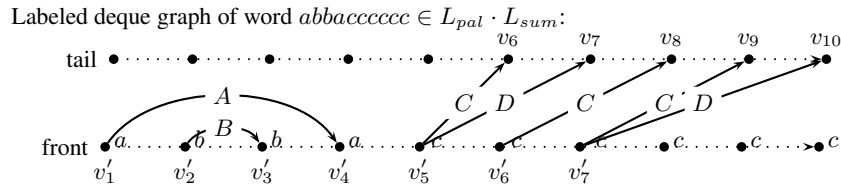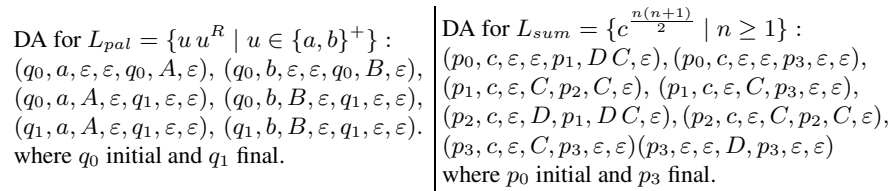
We assume that every DA M operates in *quasi-real-time* (QRT) with a delay $p \geq 1$. This means that every sequence of p moves shifts the input tape of at least one position. We say that M operates in *real time* (RT) if $p = 1$.

M starts in the initial configuration (q_0, w, ε) and accepts w if there is a sequence of moves ending in a configuration $(q, \varepsilon, \varepsilon)$ with $q \in F$, i.e, M scans from left to right the input word starting with an empty tape and accepts upon reaching a final state provided the tape is empty.

The language recognized by M, denoted by $L(M)$, is the set of words that M accepts. The family of languages recognized by a DA is denoted by $\mathcal{F}(DA)$.

Clearly, restricted types of DA correspond to QRT automata having one queue, i.e., a FIFO tape, or two stacks, i.e., two LIFO tapes with a common bottom. We illustrate Def. 1 with some basic languages.

Example 1. Any CF language can be implemented on a DA using either the front or the tail head, since the QRT condition does not restrict the recognition capability of nondeterministic PDA. We show in Fig. 1 (left) a DA using the front stack for the language of palindromes $L_{pal} = \{u\, u^R \mid u \in \{a, b\}^+\}$. Then, since the front and tail stack operations are mutually independent, it is straightforward to define a DA accepting the intersection or the shuffle product (⊔⊔) of two context-free languages. Moreover, any

DA for $L_{pal} = \{u\, u^R \mid u \in \{a, b\}^+\}$:
$(q_0, a, \varepsilon, \varepsilon, q_0, A, \varepsilon)$, $(q_0, b, \varepsilon, \varepsilon, q_0, B, \varepsilon)$,
$(q_0, a, A, \varepsilon, q_1, \varepsilon, \varepsilon)$, $(q_0, b, B, \varepsilon, q_1, \varepsilon, \varepsilon)$,
$(q_1, a, A, \varepsilon, q_1, \varepsilon, \varepsilon)$, $(q_1, b, B, \varepsilon, q_1, \varepsilon, \varepsilon)$.
where q_0 initial and q_1 final.

DA for $L_{sum} = \{c^{\frac{n(n+1)}{2}} \mid n \geq 1\}$:
$(p_0, c, \varepsilon, \varepsilon, p_1, D\, C, \varepsilon)$, $(p_0, c, \varepsilon, \varepsilon, p_3, \varepsilon, \varepsilon)$,
$(p_1, c, \varepsilon, C, p_2, C, \varepsilon)$, $(p_1, c, \varepsilon, C, p_3, \varepsilon, \varepsilon)$,
$(p_2, c, \varepsilon, D, p_1, D\, C, \varepsilon)$, $(p_2, c, \varepsilon, C, p_2, C, \varepsilon)$,
$(p_3, c, \varepsilon, C, p_3, \varepsilon, \varepsilon)(p_3, \varepsilon, \varepsilon, D, p_3, \varepsilon, \varepsilon)$
where p_0 initial and p_3 final.

Labeled deque graph of word $abbacccccc \in L_{pal} \cdot L_{sum}$:

Fig. 1. Top: DAs of Ex. 1. Bottom: labeled deque graph of a computation (explained in Sect. 3).

language accepted by a QRT QA (model named NQA_1 in [6]) is obviously accepted by a DA that implements queue operations either in the direction front-to-tail (*ft*) or tail-to-front (*tf*). Incidentally, we observe that queue operations having opposite directions are

mutually exclusive. The nonsemilinear language $L_{sum} = \{c^{\frac{n(n+1)}{2}} \mid n \geq 1\}$ is recognized by the DA in Fig. 1 (right) implementing an *ft* queue, similarly to the QAs in [6] (Stat. 3.9), which define unary languages that are the solutions of a linear homogeneous difference equation.

Then, the concatenation $L_{pal} \, L_{sum}$ is easily recognized by a DA that starts as the recognizer of L_{pal} then, upon reading c, switches to the transition relation of L_{sum}. Notice that if the deque is not empty when scanning the first c, the recognizer of $L_{pal} \, L_{num}$ rejects the input. Clearly, $L_{pal} \, L_{sum}$ cannot be recognized by a nondeterministic QRT queue automaton [6] (which cannot define the language of palindromes).

At last, a *bordered* language [8] such as $\{u \, L_{pal} \, u \mid u \in \{a, b\}^+\}$ is easily recognized: the DA non-deterministically stores the left border in the *tf* queue, then recognizes a word of L_{pal} using the front stack, then checks that the right border is identical to the stored border. Related examples inspired by the theory of DNA words [13] are possible.

From the previous examples and from the properties of QRT queue automata [6] we immediately have the following inclusions.

Theorem 1. *Family $\mathcal{F}(DA)$ strictly includes the families of languages recognized by QRT queue automata and by QRT double pushdown automata.*

Remarks. QRT double pushdown automata can easily define the shuffle product and the intersection of two CF languages. The shuffle product of two words contains all their interleavings, and, in the theory of parallel processes, represents all possible total orderings (i.e., serial scheduling) of two independent sequences of events. In our setting, such independent processes are the PDAs operating on front and on tail. In fact, a deque, viewed as a memory device, permits parallel execution of some operations, which are independent. This is in contrast with the serial behavior of, say, a PDA.

Several variants of Def. 1 are possible, differing with respect to number of states, acceptance mode, QRT constant value, richness of operations per move, and in the structure of the tape alphabet. The following variants have the same power.

Lemma 1. *Let $M = (\Sigma, \Gamma, Q, q_0, \delta, F)$ be a DA with delay $p \geq 1$ as in Def. 1.*

1. **realtime.** *There exists an RT DA equivalent to M.*
2. **stateless.** *There exists a DA with one state recognizing $L(M) \cup \varepsilon$.*
3. **simple.** *There exists a so-called* simple *DA M' equivalent to M, such that all its moves have the form $(q, a, \gamma_1, \gamma_2, q', \gamma_3, \gamma_4)$ (with $a \in \Sigma \cup \varepsilon$), where for $i = 1, 2, 3, 4$: $\gamma_i \in \Gamma \cup \varepsilon$ and at most one γ_i is not ε.*
4. **partitioned.** *There exists a DA M', equivalent to M that uses four disjoint tape symbol alphabets, Γ_{ff}, Γ_{tt}, Γ_{ft} and Γ_{tf} resp. for operations on front stack (i.e., from front to front), tail stack (i.e., from tail to tail), front-to-tail queue, and tail-to-front queue,. i.e., for every move $(q, a, \gamma_1, \gamma_2, q', \gamma_3, \gamma_4)$ of M', we have:*

$$\gamma_1 \subseteq (\Gamma_{ff} \cup \Gamma_{tf})^*, \gamma_2 \subseteq (\Gamma_{ft} \cup \Gamma_{tt})^*, \gamma_3 \subseteq (\Gamma_{ff} \cup \Gamma_{ft})^*, \text{ and } \gamma_4 \subseteq (\Gamma_{tf} \cup \Gamma_{tt})^*.$$

5. *M may have ε moves testing the deque for emptiness without increasing DA's expressive power.*

The proofs are standard and are here omitted for brevity. Space prevents detailed analysis of which normal forms can be combined together, and discussion of other accepting modes, such as acceptance by final state only. In the following, we are mainly interested in deque automata that combine the simple and the partitioned forms – they are called *SDA*. An example is in Def. 3.

Since at most one symbol is read or written in each move of an SDA, a shorthand for specifying moves (same in [14] for queue automata) is convenient. We define two copies $\overrightarrow{\Gamma}$, $\overleftarrow{\Gamma}$ of Γ as follows: for every symbol $A \in \Gamma$, a right (resp. left) pointing arrow $\overrightarrow{A} \in \overrightarrow{\Gamma}$, (resp. $\overleftarrow{A} \in \overleftarrow{\Gamma}$) means the symbol is written on the deque (resp. read from the deque); the subset of $\Gamma = \Gamma_{ff} \cup \Gamma_{ft} \cup \Gamma_{tf} \cup \Gamma_{tt}$ uniquely identifies whether the operation occurs at front or at tail. Therefore, the transition relation δ may be defined in short form as a subset of $Q \times \Sigma \times Q \times (\overrightarrow{\Gamma} \cup \overleftarrow{\Gamma})$.

For instance, the move $(q, a, \varepsilon, \varepsilon, q', A_{ft}, \varepsilon)$, which writes symbol A_{ft} on the front, is now shortened to $(q, a, q', \overrightarrow{A_{ft}})$. Similarly, $(q, a, q', \overleftarrow{A_{tf}})$, which reads A_{tf} from the front, stands for $(q, a, A_{tf}, \varepsilon, q', \varepsilon, \varepsilon)$, and $(q, a, q', \overrightarrow{A_{tf}})$ is $(q, a, A_{tf}, \varepsilon, q', \varepsilon, \varepsilon)$, i.e., A_{ft} is written to the front. Then, the transitions in short form of the DA for language L_{pal} in Fig. 1 are, without ambiguity:
$(q_0, a, q_0, \overrightarrow{A})$, $(q_0, b, q_0, \overrightarrow{B})$, $(q_0, a, q_1, \overleftarrow{A})$, $(q_0, b, q_1, \overleftarrow{B})$, $(q_1, a, q_1, \overleftarrow{A})$, $(q_1, b, q_1, \overleftarrow{B})$.
Since the DA uses the front stack, the symbols are in Γ_{ff} and should be renamed $A \equiv A_{ff}$ and $B \equiv B_{ff}$.

Closure properties of $\mathcal{F}(DA)$.

Proposition 1. *The family $\mathcal{F}(DA)$ is an AFL, but it is not a full AFL.*

The proof is standard and is omitted for brevity. Thus, family $\mathcal{F}(DA)$ enjoys a rich set of properties (e.g., closure under union, intersection with regular languages, nonerasing homomorphism, reverse homomorphism, Kleene's star, concatenation, ε-free rational transduction and inverse rational transduction), but it is not closed under erasing homomorphism. Another useful property of any AFL is closure under p-limited erasing:

Definition 2. *[7] Given a language $L \subseteq \Sigma^*$, a homomorphism h with the property that h never maps more than p consecutive symbols of any sentence $x \in L$ to ε is called a p-limited erasing on L. A class of languages is said to be closed under p-limited erasing if, for any language L of the class and any p-limited erasing h on L, $h(L)$ is in the class.*

When a homomorphism h is a p-limited erasing on L, its effect is that, when applied to any word w of L, none of the factors of w of length p is entirely erased.

2.1   Characteristic deque language and homomorphic characterization

This section introduces the language that is to DA as Dyck and AntiDyck languages resp. are to PDA and to QA. The idea is that, as for simpler data structures, deque operations can be made into a terminal alphabet containing distinct copies of each operation. Define for every $k \geq 1$, the finite alphabet Δ_k and some subsets thereof, as follows.

	$\overrightarrow{\Delta}_{ff,k}$	$\overrightarrow{\Delta}_{ft,k}$	$\overrightarrow{\Delta}_{tf,k}$	$\overrightarrow{\Delta}_{tt,k}$
$\overrightarrow{\Delta}_k$	$\overrightarrow{ff}_1, \ldots, \overrightarrow{ff}_k$	$\overrightarrow{ft}_1, \ldots, \overrightarrow{ft}_k$	$\overrightarrow{tf}_1, \ldots, \overrightarrow{tf}_k$	$\overrightarrow{tt}_1, \ldots, \overrightarrow{tt}_k$
$\overleftarrow{\Delta}_k$	$\overleftarrow{ff}_1, \ldots, \overleftarrow{ff}_k$	$\overleftarrow{ft}_1, \ldots, \overleftarrow{ft}_k$	$\overleftarrow{tf}_1, \ldots, \overleftarrow{tf}_k$	$\overleftarrow{tt}_1, \ldots, \overleftarrow{tt}_k$
	$\overleftarrow{\Delta}_{ff,k}$	$\overleftarrow{\Delta}_{ft,k}$	$\overleftarrow{\Delta}_{tf,k}$	$\overleftarrow{\Delta}_{tt,k}$

The alphabet is $\Delta_k = \{\overrightarrow{ff}_1, \ldots, \overleftarrow{tt}_k\}$. Thus, the set of "open brackets" in Δ_k is denoted by $\overrightarrow{\Delta_k}$, the set of closing brackets of the form $\overleftarrow{ff}_j$ as $\overleftarrow{\Delta}_{ff,k}$, etc.

The natural definition of CDL is by means of a deterministic RT DA with just one state, in analogy with the PDA recognizing the Dyck language.

Definition 3. *For each $k \geq 1$, the* characteristic deque language *(CDL), denoted by $DQ_k \subset \Delta_k^*$, is the language accepted by the SDA*

$$A_{CDL_k} = (\{q_0\}, \Delta_k, \Gamma, \delta, q_0, \{q_0\})$$

where $\Gamma = \bigcup_{1 \leq j \leq k}(\{\, FF_j,\ TF_j,\ TT_j,\ FT_j\})$ and the transition relation is defined as follows (the only state of A_{CDL_k} is omitted):

$$\delta = \bigcup_{1 \leq j \leq k} \left\{ \begin{array}{l} (\overrightarrow{ff}_j, \overrightarrow{FF}_j), (\overrightarrow{ft}_j, \overrightarrow{FT}_j), (\overrightarrow{tf}_j, \overrightarrow{TF}_j), (\overrightarrow{tt}_j, \overrightarrow{TT}_j), \\ (\overleftarrow{ff}_j, \overleftarrow{FF}_j), (\overleftarrow{ft}_j, \overleftarrow{FT}_j), (\overleftarrow{tf}_j, \overleftarrow{TF}_j), (\overleftarrow{tt}_j, \overleftarrow{TT}_j) \end{array} \right\}.$$

Notice that this machine is deterministic, real time and "stateless". To illustrate, word $\overrightarrow{tt}_1\overrightarrow{ff}_1\overrightarrow{tt}_2\overleftarrow{ff}_1\overrightarrow{ft}_1\overleftarrow{tt}_2\overrightarrow{ft}_2\overleftarrow{tt}_1\overleftarrow{ft}_1\overleftarrow{ft}_2 \in DQ_2$ is accepted with the computation:

$$\varepsilon \xrightarrow{\overrightarrow{tt}_1} TT_1 \xrightarrow{\overrightarrow{ff}_1} FF_1\, TT_1 \xrightarrow{\overrightarrow{tt}_2} FF_1\, TT_1\, TT_2 \xrightarrow{\overleftarrow{ff}_1} TT_1\, TT_2 \xrightarrow{\overrightarrow{ft}_1} TT_1\, TT_2\, FT_1$$

$$\xrightarrow{\overleftarrow{tt}_1} TT_2\, FT_1 \xrightarrow{\overrightarrow{ft}_2} TT_2\, FT_1\, FT_2 \xrightarrow{\overleftarrow{tt}_2} FT_1\, FT_2 \xrightarrow{\overleftarrow{ft}_1} FT_2 \xrightarrow{\overleftarrow{ft}_2} \varepsilon$$

In words, the sentences of CDL are the sequences that obey the natural schedule of deque operations. Other characterizations of CDL in terms of graphs, cancellation rules and shuffles will be given in Sect. 3.

Using CDL we characterize deque languages *à la* Chomsky-Schützenberger.

Theorem 2. *A language $L \subseteq \Sigma^*$ is accepted by a deque automaton if, and only if, there exist $k > 0$, $p > 0$, a finite alphabet Θ, a homomorphism $g : \Theta \to \Delta_k^*$, a regular language R on Θ, and a p-limited erasing $h : \Theta \to \Sigma \cup \varepsilon$ on R such that $L = h\left(g^{-1}(DQ_k) \cap R\right)$.*

Proof. By the closure properties of an AFL it is obvious that if $L = h\left(g^{-1}(DQ_k) \cap R\right)$ for some h, g, R, k verifying the statement of the theorem, then L can be recognized by a DA. Let now L be recognized by a DA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, which we assume to be in simple partitioned normal form. Therefore, $\delta \subseteq Q \times (\Sigma \cup \varepsilon) \times Q \times (\overrightarrow{\Gamma} \cup \overleftarrow{\Gamma})$, where $\overrightarrow{\Gamma}$ and $\overleftarrow{\Gamma}$ are two disjoint copies of the tape alphabet Γ.

Since Γ is partitioned in Γ_{ft}, Γ_{tt}, etc., the alphabet $\overrightarrow{\Gamma} \cup \overleftarrow{\Gamma}$ can be considered as a characteristic alphabet Δ_k, with $k = |\Gamma|$. Define the finite alphabet $\Theta = \delta$, i.e., Θ is the set of all quadruples in δ. Let $g : \Theta \to \Delta_k$ be the homomorphism defined by $g(\langle q, a, q', A \rangle) = A$, for every $q, q' \in Q, a \in \Sigma \cup \varepsilon, A \in \overrightarrow{\Gamma} \cup \overleftarrow{\Gamma}$. Let R be

the local language in Θ^* defined by the pairs of consecutive transitions of M (e.g., $\langle q, a, q', \overleftarrow{A}\rangle\langle q', b, q'', \overleftarrow{B}\rangle$, etc.)

Let h be the projection of Θ on the 2nd component, i.e., $h(\langle q, a, q', A\rangle) = a$. It is obvious that $h\left(g^{-1}(DQ_k) \cap R\right)$ is L. Homomorphism h is p-erasing for R and thus for the language $g^{-1}(DQ_k) \cap R$, where p is the QRT constant of M, since it returns ε only in correspondence to ε-transitions of M. $\square$

By a standard procedure (for QA in [14]), in the statement of Theor. 2 it is possible to assume $k = 2$. In fact, let $\rho : \Delta_k \to \Delta_2^+$ be the homomorphism defined by $\rho(a_j) = a_1 a_2^j$, for every $a_1, a_2, a_j \in \overrightarrow{\Delta_k}$, and $\rho(a_j) = a_2^j a_1$, for every $a_1, a_2, a_j \in \overleftarrow{\Delta_k}$. Then $\rho^{-1}(DQ_2) = DQ_k$. $\square$

2.2   Example of expressiveness of deque languages

Deque automata have a noteworthy capability to define languages that replicate, any number of times, a factor or its reversal. We can introduce a schema for specifying replications, by means of a regular language $\Pi \subseteq \{D, R\}^*$ where D and R resp. stand for "direct" and "reverse". Intuitively, a word such as $DDRD$ specifies that a given word $u \in \Sigma^+$ is followed by 4 replicas: uuu^Ru. We can define languages parameterized by a replication schema, e.g., $L_{(D^*RDR)} = \left\{u\,u^*\,u^R\,u\,u^R \mid u \in \Sigma^+\right\}$. A family $\{L_{(\Pi)}\}$, of languages parameterized by a replication schema Π, called *regular replica family*, is defined as follows, using the following family of homomorphisms, for every $u \in \Sigma^+$: $\rho_u : \{D, R\} \to \Sigma^+$ is defined as $\rho_u(D) = u, \rho_u(R) = u^R$. Then the language is

$$L_{(\Pi)} = \{w \in \Sigma^+ \mid \exists u \in \Sigma^+, \pi \in \Pi : w = \rho_u(\pi)\}.$$

Proposition 2. *For each regular replication schema $\Pi \subseteq \{D, R\}^*$, the language $L_{(\Pi)}$ is in $\mathcal{F}(DA)$.*

Proof. (hint) Let $L = \{w \in (\Sigma \cup \{D, R\})^+ \mid w = u(Du \cup Ru^R)^+, u \in \Sigma^+\}$. We claim that L is in $\mathcal{F}(DA)$, hence also $L_{(\Pi)}$ is in $\mathcal{F}(DA)$: let Π' be the regular language obtained by the shuffle of Π with Σ^+; by closure under intersection with regular languages also $L \cap \Pi'$ is in $\mathcal{F}(DA)$; by applying an obvious 1-limited erasing we obtain $L_{(\Pi)}$. In the initial state q_0, given a word $uD \ldots$ or $uR \ldots$, M stores factor u in the deque at the tail's end, so that when reading from the front the deque content is u, when reading from the tail it is u^R. Upon reading D or R, M changes its state to the state q_F or, resp., to the state q_T and writes a new symbol Z to the tail or, resp., to the front; the content of the deque is uZ or, resp., Zu. In state q_F, M compares the current input symbol a with the symbol at the front and writes a to the tail. When D or R are scanned and Z is read from the front, M changes its state to q_F (if D) or q_T (if R) and writes Z to the tail or resp. to the front; the deque content is still uZ or resp. Zu. The behavior in state q_T is symmetrical: M compares the input with u^R read from the tail and rewrites it to the front, so that at the end the tape content is still u. M guesses when the current factor is the last one and then ceases to store the replica on the tape. $\square$

3   Characterization of deque sentences by planar graphs

An insightful analysis of deque operation sequences has been recently obtained within
research on graph drawing by Auer et al. [2, 1]. Given a planar graph (V, E) having
exactly one hamiltonian path, imagine to draw the vertices V so that the hamiltonian
path edges, called E_H, lay on a straight line, called a *linear layout*, which thus visual-
izes a total order on V. The other edges $E - E_H$ can be drawn without crossing. The
direction of edge (v_i, v_j), $i < j$, is $v_i \rightarrow v_j$. Such a drawing is called a *deque graph* be-
cause each non-hamiltonian edge represents two operations that resp. insert and remove
an item from the deque. Extending previous work on the linear layout of simpler data
structures (one/two pushdown stacks, queue), Auer et al. [2, 1] have precisely charac-
terized the deque graphs. We informally introduce the concepts from the cited studies
that are relevant for studying deque automata and languages.

   A deque graph represents a sequence of deque operations: writing occurs at the ori-
gin v_i of a (non-hamiltonian) edge $v_i \rightarrow v_j$ and reading at the destination v_j. Since the
hamiltonian path totally orders vertices, if we label each vertex with a terminal char-
acter, we obtain a word. Fig. 2, derived from [2], shows (top) a planar deque graph,
(bottom) its 3D representation on a torus or cylinder, and (middle) its plane layout ob-
tained by cutting the torus surface along the hamiltonian path and duplicating vertices.
Conceptually, the primary representation in [2, 1] is the graph drawn on torus, called
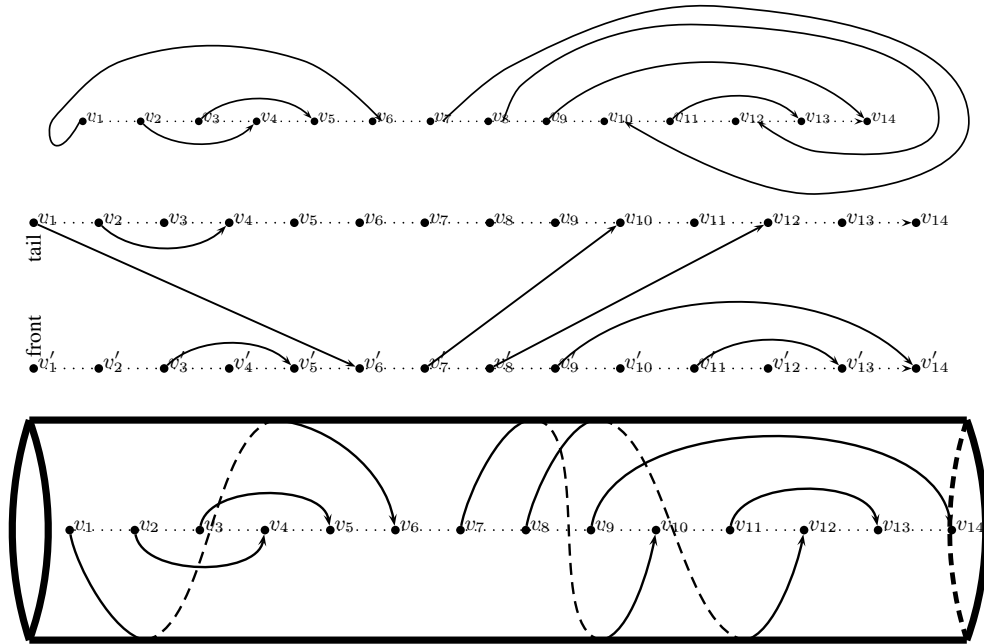
Fig. 2. Top: planar graph with hamiltonian path dotted. Middle: the same graph represented as
deque graph with duplicated vertices. Bottom: 3D linear cylindric drawing of the same graph.
The middle graph is achieved by cutting along the hamiltonian path and unrolling the cylinder
surface, and duplicating vertices.

a *linear cylindric drawing* (LCD). The main result is that an LCD represents a valid sequence of deque operations, and is called a *deque graph*, if, and only if, it is a planar graph with one hamiltonian path. For practicality the plane layout (Fig. 2, middle) is preferred, but an LCD would be more appropriate as a model of 3D-sequences, such as (bio)chemical strings of molecules. To transform an LCD into a deque graph, the hamiltonian path is duplicated and each vertex v_i is represented by a pair of points v_i, v_i', resp. placed on the upper and lower path, which resp. represent the deque tail and front sides. We now introduce a labeling of the vertices, to associate a deque graph with a word.

Definition 4. *Let Σ be a terminal alphabet and let $G = (V, E)$ be a deque graph. A* labeled deque graph *(LDG) is defined by the pair (G, λ) where λ is the* labeling function $\lambda : V \to \Sigma$, *which assigns a terminal character to each vertex. The* word defined *by an LDG (G, λ), denoted by $W(G, \lambda)$, is $\lambda(v_1)\lambda(v_2)\ldots\lambda(v_n)$, where $v_1 v_2 \ldots v_n$ is the hamiltonian path. Let $\mathcal{G} = \{\ldots, (G_m, \lambda), \ldots\}$ be a family of LDGs. The* language defined by the graph family *is $W(\mathcal{G}) = \bigcup_{(G_m, \lambda_m) \in \mathcal{G}} W(G_m, \lambda_m)$.*

Definition of CDL by means of graphs. Using LDGs, we study the sequences of deque operations which may occur in a CDL, i.e., the sentences of language DQ_k of Def. 3. We only consider deque graphs such that exactly one deque operation occurs at each vertex, i.e., one non-hamiltonian edge impinges on it. Auer's model allows any number of non-hamiltonian edges per vertex, but here we want to ensure that the label of each vertex carries enough information about the type of the incoming/outgoing edge.
There are four edge types depending on where their source and destination vertices lie on the unrolled cylinder: front-to-front (*ff*), tail-to-tail (*tt*), front-to-tail (*ft*) and tail-to-front (*tf*). The type of an edge e is denoted by $\tau(e)$. Next, we label the vertices with "'brackets'" that carry the information about the type of edge they belong to.

Definition 5. *Let $G = (V, E)$ be a deque graph. For each $k \geq 1$, we denote by $\mathcal{G}_k$ the family of all (G, λ), where the* characteristic labeling function $\lambda : V \to \Delta_k$ *is defined as follows. For each edge $e = (v_i \to v_j) \in E$, $\exists 1 \leq h \leq k$ and*

if $\tau(e) = ff$ then $\lambda(v_i) = \overrightarrow{ff}_h$; $\lambda(v_j) = \overleftarrow{ff}_h$ $\|$ if $\tau(e) = tt$ then $\lambda(v_i) = \overrightarrow{tt}_h$; $\lambda(v_j) = \overleftarrow{tt}_h$

if $\tau(e) = ft$ then $\lambda(v_i) = \overrightarrow{ft}_h$; $\lambda(v_j) = \overleftarrow{ft}_h$ $\|$ if $\tau(e) = ft$ then $\lambda(v_i) = \overrightarrow{tf}_h$; $\lambda(v_j) = \overrightarrow{tf}_h$.

Example 2. Fig. 3 show an LDG based on the characteristic labeling over Δ_2 of the deque graph in Fig. 2 (middle). The word defined by the graph is:
$\overrightarrow{tf}_2 \overrightarrow{tt}_1 \overrightarrow{ff}_1 \overleftarrow{tt}_1 \overleftarrow{ff}_1 \overrightarrow{tf}_2 \overrightarrow{ft}_1 \overrightarrow{ft}_1 \overrightarrow{ff}_2 \overleftarrow{ft}_1 \overrightarrow{ff}_1 \overleftarrow{ft}_1 \overleftarrow{ff}_1 \overleftarrow{ff}_2$. The deque content after reading the prefix $\overrightarrow{tf}_2 \overrightarrow{tt}_1 \overrightarrow{ff}_1 \overleftarrow{tt}_1$ is $TF_2 FF_1$, which in Fig. 3 is visible as the vertical cut of edges (v_1, v_6') and (v_3', v_5').

We show that each CDL is the language defined by a family of characteristic LDGs.

Theorem 3. *The following identity holds: $W(\mathcal{G}_k) = DQ_k$.*

The proof, omitted for brevity, first makes precise the definition of deque machines of [1], that we call *Deque Graph Machines* (DGMs), and we enrich with a finite alphabet to label vertices, not present in [1], so that it can be seen as accepting words, in particular over the characteristic alphabet. Such machine is called a Δ_k-DGM. When
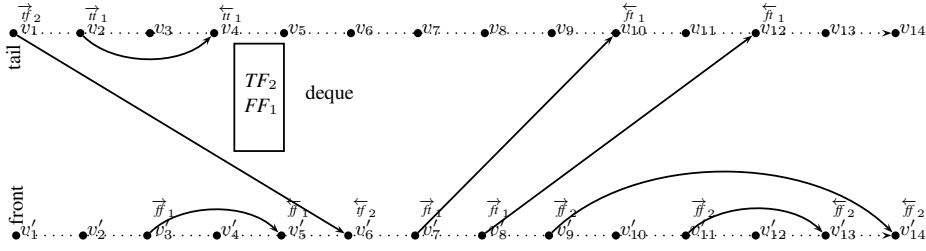
Fig. 3. A characteristic labeling over Δ_2 of the deque graph in Fig. 2 (middle). The deque content after four steps is shown.

storing an edge $e = \langle \alpha, \beta \rangle$ of an LDG (G, λ), with $G = (V, E)$, also the label of the destination vertex β must be stored; the label is denoted, with an abuse of notation, by $\lambda(e)$ (with $\lambda : E \to \overleftarrow{\Delta_k}$).

The result is proved by showing that, when reading the same prefix y of a word yx in DQ_k, the configurations $\langle x, q_0, u \rangle$ of A_{CDL_k} and $\langle x, q_0, v \rangle$ of Δ_k-DGM are such that $\Lambda(v) = u$, where $\Lambda : E \to \Gamma$ is the homomorphism defined as the "upper case" version of λ, i.e., if $\lambda(e) = \overleftarrow{ff}_j$ then $\Lambda(e) = FF_j$, etc.

Definition of characteristic deque languages via cancellation rules. The well known definition of Dyck's languages as the equivalence class of the words obtained by applying the cancellation rule $(\,) \to \varepsilon$ is now combined with the similar definition of the AntiDyck languages [14] into a set of cancellation rules defining DQ_k languages.

Theorem 4. *A word over alphabet Δ_k, $k \geq 1$, is in language DQ_k if, and only if, the application zero or more times (in any order) of the following* cancellation rules *(CR) reduces the word to empty. For all $v \in \Delta_k^{\,*}$:*

1. $\forall u \in (\Delta_k - \overleftarrow{\Delta}_{ff,k})^*, \forall x \in (\overleftarrow{\Delta}_{ft,k} \cup \overrightarrow{\Delta}_{tf,k} \cup \Delta_{tt,k})^* : u \,\overrightarrow{ff}_i\, x \,\overleftarrow{ff}_i\, v \to uxv$
2. $\forall u \in (\Delta_k - \overleftarrow{\Delta}_{tt,k})^*, \forall x \in (\Delta_{ff,k} \cup \overrightarrow{\Delta}_{ft,k} \cup \overleftarrow{\Delta}_{tf,k})^* : u \,\overrightarrow{tt}_i\, x \,\overleftarrow{tt}_i\, v \to uxv$
3. $\forall x \in (\overrightarrow{\Delta}_{ft,k})^* : \overrightarrow{ft}_i\, x \,\overleftarrow{ft}_i\, v \to xv$
4. $\forall x \in (\overrightarrow{\Delta}_{tf,k})^* : \overrightarrow{tf}_i\, x \,\overleftarrow{tf}_i\, v \to xv$.

The proof (omitted for brevity) shows that, for any word accepted by A_{CDL_k}, each cancellation step preserves correctness, and conversely.

Composition of Dyck and AntiDyck. CRs can be applied in the following order, repeatedly scanning the input word from left to right: CR 1 and 2 delete all Dyck symbols; then, CR 3 and 4 all AntiDyck symbols. CR 1, 2 could be reformulated so that an application of either rule is not necessarily leftmost. Such remarks suggest to express CDL as a suitable combination of Dyck and AntiDyck languages. Define two CF languages and a $\mathcal{F}(DA)$ language, using the Dyck languages $Dyck_{ff}$, $Dyck_{tt}$, resp. over $\Delta_{ff,k}$ and $\Delta_{tt,k}$, and the AntiDyck ($Adyck$) languages over $\Delta_{ft,k}$ and $\Delta_{tf,k}$:

$$H_1 = \left(Dyck_{ff} \cdot (\overrightarrow{\Delta}_{ft,k} \cup \overleftarrow{\Delta}_{tf,k})^*\right)^* Dyck_{ff} \quad \bigg| \quad H_3 = \phi^{-1}\left((Adyck_{ft} \cup Adyck_{tf})^*\right)$$

$$H_2 = \left(Dyck_{tt} \cdot (\overleftarrow{\Delta}_{ft,k} \cup \overrightarrow{\Delta}_{tf,k})^*\right)^* Dyck_{tt} \quad \bigg| \quad \text{with } \begin{cases} \phi(a) = \varepsilon, \ a \in \Delta_{ff,k} \cup \Delta_{tt,k} \\ \phi(a) = a, \ a \in \Delta_{ft,k} \cup \Delta_{tf,k} \end{cases}$$

Theorem 5. *For each CDL the identity holds:* $DQ_k = (H_1 \uplus H_2) \cap H_3$.

Proof. Let $H_4 = (H_1 \uplus H_2) \cap H_3$. Claim $H_4 \subseteq DQ_k$: we show that if $w' \in H_4$, w' reduces to ε via CR, hence it is in DQ_k. By definition of shuffle, $w' \in H_1 \uplus H_2$ has the form $x' \overrightarrow{ff}_j u' \overleftarrow{ff}_j v'$ where x' does not contain any closed parenthesis of $Dyck_{ff}$, the symbols $\overrightarrow{ff}_j$, $\overleftarrow{ff}_j$ form the innermost pair in Dyck_{ff} and the factor u' does not contain $\overleftarrow{tf}_i$ or $\overrightarrow{ft}_i$. Therefore, CR 1 applies over and over until all symbols in $\Delta_{ff,k}$ are canceled. Similarly, CR 2 cancels all parentheses in $Dyck_{tt}$. Then w' is reduced to a word w'' over AntiDyck symbols. Since $w'' \in H_3$, CR 3 and CR 4 reduce it to ε. Claim $DQ_k \subseteq H4$: by Theor. 3, there exists an LDG (G, λ) such that $w \in W(\mathcal{G})$. Consider the sequence w_f of labels on the front row of (G, λ) (e.g., in Fig. 3 the subgraph with vertices $v'_1, \ldots, v'_n$). Clearly, word w_f is over alphabet $\Delta_{ff,k} \cup \{\overrightarrow{ft}_1, \overleftarrow{tf}_1, \ldots, \overleftarrow{tf}_k, \}$ and belongs to language H_1. Similarly, the sequence of labels on the tail row (i.e., on vertices $v_1, \ldots, v_n$) is a word w_t in language H_2. Notice that $|w_f| + |w_t| = n$ and $w = w_f \uplus w_t$, i.e., $w \in H_1 \uplus H_2$. It remains to prove that $w \in H_3$. This is immediate since the subgraph labeled with the projection of w over $\Delta_{ft,k} \cup \Delta_{tf,k}$ contains all and only the queue edges, which do not cross and satisfy the definition of H_3. $\square$

Corollary 1. *For all languages L in family $\mathcal{F}(DA)$, there exist context-free languages L_1, L_2, a language L_Q recognized by a QRT queue automaton, and a rational transduction τ such that $L = \tau\left((L_1 \uplus L_2) \cap L_3\right)$.*

Linear deque graphs for generic deque languages. Since each CDL word is represented by an LDG (Theor. 3) and from Theor. 2 each word of a language in $\mathcal{F}(DA)$ is the homomorphic image of a CDL word, we can equip any deque language with a valuable graphical representation similar to CF syntax trees but more general.

We loosely describe for Ex. 1 how the computation of $abbacccccc \in L_{pal}L_{sum}$ is converted into the LDG shown in Fig. 1. For clarity, we denote A with A_{ff}, B with B_{ff}, etc. Scanning the first a, the DA pushes A_{ff}. The following moves are: scan b and push B_{ff}, scan a and pop B_{ff}. The four preceding moves are represented by edges (v'_1, v'_4) and (v'_2, v'_3) which compose a so-called "rainbow" pattern, essentially isomorphic to a syntax tree. Now the configuration is $(q_1, ccccc, \varepsilon)$ and the next moves are: scan c and write $C_{ft}D_{ft}$; scan c, read C and write C_{ft}; scan c, read D and write $C_{ft}D_{ft}$. Now the configuration is $(p_3, ccc, C_{ft}C_{ft}D_{ft})$, the DA scans ccc and empties the tape. To be precise, the two edges exiting v'_5 (and also v'_7) represent a non-simple DA transition. Such transitions where not present in A_{CDL}, and its LDG did not have vertices with non-hamiltonian degree 2 (already observed). But, it would be immediate to normalize the DA in Fig. 1, by introducing, between v'_5 and v'_7, a new vertex v''_5 with label ε and replacing edge (v'_5, v'_7) with edge (v''_5, v'_7). The latter edge is associated with an ε-transition, and corresponds to an erasing value of the homomorphism of Theor. 2.

We have seen that all $\mathcal{F}(DA)$ languages have a syntax structure representable by toroidal embedding of planar graphs having one hamiltonian path. In such LDG graphs

it is possible to recognize subgraphs corresponding to Dyck languages (*ff* and *tt*) and to AntiDyck languages (*ft* and *tf*); an illustration is the regular replicas language defined in Prop. 2. Evidence for similar patterns occurs in the linear representation of some models for chemical and biological molecular structures (e.g., [4, 13]). It is known that some of those patterns also occur in natural languages and are to some extent captured by existing grammar models, in particular by *dependency grammars*.

Conclusion. We have shown that a deque used as memory of an automaton, permits computations beyond those possible with stacks and queues and defines an interesting language family. Such computations are nicely represented by planar toroidal graphs, which facilitate intuitive reasoning on deque languages. The basic closure and inclusion properties for deque automata have been established, but much remains to be done, in particular concerning deterministic deque machines. Comparisons with other existing models of grammars and automata featuring stacks and queues remain for the future.
Acknowledgements. We thank the anonymous reviewers for their useful suggestions.

References

1. C. Auer. *Planar graphs and their duals on cylinder surfaces*. PhD thesis, Univ. Passau, 2014.
2. C. Auer, C. Bachmaier, F. Brandenburg, W. Brunner, and A. Gleißner. Plane drawings of queue and deque graphs. In U. Brandes and S. Cornelsen, editors, *Graph Drawing - 18th Int. Symp.*, pages 68–79. Springer, 2011.
3. K. Ayers. Deque automata and a subfamily of context-sensitive languages which contains all semilinear bounded languages. *Theor. Comput. Sci.*, 40:163–174, 1985.
4. S. Barthel and D. Buck. Toroidal embeddings of abstractly planar graphs are knotted or linked. *Journal of Mathematical Chemistry*, 53(8):1772–1790, 2015.
5. F. Brandenburg. A note on: 'deque automata and a subfamily of context-sensitive languages which contains all semilinear bounded languages'. *Theor. Comput. Sci.*, 52:341–342, 1987.
6. A. Cherubini, C. Citrini, S. Crespi-Reghizzi, and D. Mandrioli. QRT FIFO automata, breath-first grammars and their relations. *Theor. Comput. Sci.*, 85(1):171–203, 1991.
7. J. Hopcroft and J. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley, Wokingham, 1979.
8. L. Kari and M. S. Kulkarni. Disjunctivity and other properties of sets of pseudo-bordered words. *Acta Inf.*, 54(4):379–398, 2017.
9. D. E. Knuth. *The Art of Computer Programming, Volume I: Fundamental Algorithms*. Addison-Wesley, 1968.
10. M. Kutrib, A. Malcher, and M. Wendlandt. Queue automata: Foundations and developments. In A. Adamatzky, editor, *Reversibility and Universality: Essays Presented to Kenichi Morita on the Occasion of his 70th Birthday*, pages 385–431. Springer, 2018.
11. B. L. Leong and J. I. Seiferas. New real-time simulations of multihead tape units. *J. ACM*, 28(1):166–180, 1981.
12. H. Petersen. Stacks versus deques. In J. Wang, editor, *Computing and Combinatorics, COCOON 2001*, pages 218–227. Springer, 2001.
13. D. B. Searls. The language of genes. *Nature*, 420(6912):211–217, 2002.
14. B. Vauquelin and P. Franchi-Zannettacci. Automates a file. *Theor. Comput. Sci.*, 11:221–225, 1980.
15. R. Vollmar. Über einen Automaten mit Pufferspeicherung. *Computing*, 5(1):57–70, 1970.