The Thirty-First International Florida
Artificial Intelligence Research Society Conference (FLAIRS-31)

Towards Foundations of Agents Reasoning on Streams of Percepts

Özgür L. Özçep, Ralf Möller
Institute of Information Systems (IFIS)
University of Lübeck
Lübeck, Germany
{oezcep,moeller}@ifis.uni-luebeck.de

Abstract

Mapping percepts to actions is at the heart of the agents metaphor. However, little work has investigated this mapping from a stream-based perspective. Inspired by previous work on the foundations of stream processing, we analyze properties of window-based percept processing and identify properties of stream-processing agents with representation theorems. The resulting axiomatizations help us to deepen the understanding of agents that can be safely implemented.

1 Introduction

Rational agents (Russell and Norvig 1995) acting in a dynamic environment have to be equipped with stream processing capabilities for streams of different types and levels of inputs. These may be streams of percepts, but also streams of internal representations produced from lower-level streams: streams of symbolic representations of percepts, of agent's beliefs on the state of the world, of actions be carried out or decisions to be taken in order to reach a goal etc. Moreover, in many cases the agent has to process not a single stream but multiple streams that have to be combined and cascaded, thereby producing streams of a possibly different type. In the setting we consider, the stream capabilities are represented as stream queries, i.e. functions mapping (one or more) input streams to an output stream. A simple example stream query Q is one outputting every time a new element arrives the last two elements of the input-stream. So, e.g., Q would map the stream of letters $a, b, c, d, e, f \ldots$ to $ab, bc, cd, de, ef \ldots$.

The kind of "stream reasoning" we consider here is basic reasoning on streams of percepts w.r.t. two main challenging aspects: the potential infinity of streams and the policy that the stream has to be processed in a one-pass fashion using small time and space resources. This basic reasoning is a prerequisite to realize higher-order reasoning on plans where the next best action is chosen (w.r.t. some cost function) on the basis of the percept streams and other internal streams.

The usual solution to handle the potential infinity is to apply a window function, i.e., a function which grabs only a finite portion of the stream at every time point. A formal justification for the use of windows can be derived from a result of Gurevich and colleagues (Gurevich, Leinders, and Van Den Bussche 2007) which states that all stream functions depending only on finite prefixes of the input can be defined as an iterative application of a window function—and vice versa. Following the foundational work of Gurevich and others, but emphasizing the axiomatic and representational aspects, this paper investigates stream processing from an input-output perspective—formalizing the stream capabilities as stream queries. This enables a formal description of the properties that an agent's reasoning engine should posses by axioms constraining its input-output behaviour.

We consider different properties a stream query might or should have and state them as axioms. Then we show which stream queries actually fulfill these axioms. In particular we show: Stream queries that are defined on infinite streams only—and not on both, finite and infinite streams as in (Gurevich, Leinders, and Van Den Bussche 2007)—and that are prefix-determined can be represented as iterative applications of a window function. In this general result, the window function is allowed to refer to the whole stream history. But this is not in accordance with the one-pass and small space policy. Hence, as a special case, we also investigate stream queries induced by constant-width windows and show how to characterize them axiomatically with a generalization of the distribution property.

The axiomatic characterizations given in this paper are on a basic phenomenological level—phenomenological, because only observations regarding the input-output behavior are taken into account, and basic, because no further properties regarding the structure of the data stream elements are presupposed. However, an axiomatic understanding of this basic form of stream reasoning lays the ground for an ambitious axiomatic characterization of rational agents where also the properties of various higher-order streams of states such as beliefs or goals are taken into account.

The paper is structured as follows: After this introduction we discuss related work in comparison to ours. In Section 3 we introduce necessary technical terminology. The main results of the paper are developed in Section 4, which describes axiomatic characterizations of genuine stream function, stream functions based on constant-size windows, and stream functions respecting time constraints. Section 5 gives a short summary and an outlook.

2 Related Work

The work presented here is based on the foundation of stream processing according to (Gurevich, Leinders, and Van Den Bussche 2007) which considers streams as finite or infinite words. The research on streams from the word perspective is quite mature and the literature on infinite words, language characterizations, and associated machine models abounds. The focus in this paper is on representational aspects for functions from words to words. For all other interesting topics and relevant research papers on infinite words we refer the reader to (Perrin and Pin 2004).

The approach of this paper is axiom based. An example of an axiom-based approach for stream processing is given in (Rabinovich 2003), but in (Rabinovich 2003) the emphasis is on temporal streams with a continuous time domain.

In this paper, streams are considered in an abstract way, with stream elements over an arbitrary domain D. In higher-level stream processing, where the domain elements have a certain semantics according to specifications in a knowledge base or an ontology (as in ontology-based stream access) further aspects related to semantics become relevant: correct window semantics w.r.t. the given domain semantics (Özçep, Möller, and Neuenstadt 2014), reasoning aspects (Heintz, Kvarnström, and Doherty 2010) or equivalence of stream queries (Beck, Dao-Tran, and Eiter 2016) etc.

We consider the aspect of performant processing on streams in the context of constant-width windows. In (Arasu et al. 2004) syntactical criteria for memory-boundedness in data-stream management systems are considered.

The function-oriented consideration of stream queries in this papers lends itself to a pipeline-style functional programming language on streams. One approach showing the practical realizability of such a programming language is described in (Cowley and Taylor 2011).

3 Preliminaries

We use the following simple definition of streams of words over an alphabet D. We call alphabets D also *domains*.

Definition 1. *The set of* finite streams *is the set of finite words D^* over the alphabet D. The set of* infinite streams *is the set of ω-words D^ω over D. The set of (all) streams is denoted $D^\infty = D^* \cup D^\omega$.*

The basic definition of streams above is general enough to capture all different forms of streams (in particular temporal streams) that are considered in the approaches mentioned in the section on related work.

$D^{\leq n}$ is the set of words of length maximally n. For any finite stream s the length of s is denoted by $|s|$. For infinite streams s let $|s| = \infty$ for some fixed object $\infty \notin \mathbb{N}$. For $n \in \mathbb{N}$ with $1 \leq n \leq |s|$ we let $s^{=n}$ be the n-th element in the stream s. For $n = 0$ let $s^{=n} = \epsilon =$ the empty word. $s^{\leq n}$ denotes the n-prefix of s, $s^{\geq n}$ is the suffix of s s.t. $s^{\leq n-1} \circ s^{\geq n} = s$. For an interval $[j,k]$, with $1 \leq j \leq k$, $s^{[j,k]}$ is the stream of elements of s such that $s = s^{\leq j-1} \circ s^{[j,k]} \circ s^{\geq k+1}$. For a finite stream $w \in D^*$ and a set of streams X the term $w \circ X$ or shorter wX denotes the set of all w-extensions with words from X: $\{s \in D^\infty \mid$ There is $s' \in X$ s.t. $s = w \circ s'\}$. The finite word s is a prefix of a word s', for short $s \sqsubseteq s'$, iff there is a word v such that $s' = s \circ v$. If $s \sqsubseteq s'$, then $s' -_{\sqsubseteq} s$ is the suffix of s' when deleting its prefix s. If all letters of s occur in s' in the ordering of s (but perhaps not directly next to each other) then s is called a *subsequence* of s'. If $s' = usv$ for $u \in D^*$ and $v \in D^\infty$, then s is called a subword of s'. We are going to write streams in the word notation, sometimes mentioning the concatenation $\circ$ explicitly. For a function $Q : D_1 \longrightarrow D_2$ and $Y \subseteq D_2$ let $Q^{-1}[Y] = Q^{-1}(Y) = \{w \in D_1 \mid Q(w) \in Y\}$.

4 Streams in the (Infinite) Word Perspective

The main aim of this paper is to axiomatically characterize different classes of functions of the form $Q : D_1^\infty \longrightarrow D_2^\infty$. So the focus is on total functions which map a finite or infinite stream over a given domain D_1 to a finite or infinite stream on a domain D_2. In this paper we are going to assume without loss of generality the same domain $D = D_1 = D_2$ for inputs and outputs. All functions of this form will be named "stream queries" and will be denoted by Q or primed and indexed variants of Q.

By considering the union of finite and infinite streams as potential domains and ranges of stream queries we are following the approach of (Gurevich, Leinders, and Van Den Bussche 2007). Later we also consider—thereby following (Weihrauch 2000)—functions where the domain (resp. the range) is either the set of finite streams D^* or the set of infinite streams D^ω. The more general definition of a stream query according to Gurevich and colleagues allows to cover different scenarios that have found interest in Computer Science. In particular, allowing for finite streams as potential inputs of stream queries allows to cover scenarios where an agent is allowed to stop processing a query after a finite number of steps. It also covers the case where the agent gets informed about the fact that the input stream is finite: You can stop processing because there is no element to come anymore. A framework for implementing such "informed" stream processing is developed under the term "punctuation semantics" in (Tucker et al. 2003).

We note that in the framework of (Gurevich, Leinders, and Van Den Bussche 2007) *multiple streams* can be handled by attaching to the domain elements tags with provenance information, in particular information on the stream source from which the element originates. This tag-approach is too simple in the sense that there is no control on how to interleave the stream inputs—as is, e.g., the case in state-of-the art stream query languages following a pipeline architecture. But the framework of (Gurevich, Leinders, and Van Den Bussche 2007) can be extended to handle functions of the form $Q : D^\infty \times \cdots \times D^\infty \longrightarrow D^\infty$ similar to the approach of (Weihrauch 2000).

4.1 Genuine Stream Queries are Window-Based

Though any function $Q : D^\infty \longrightarrow D^\infty$ is termed a stream query, only those queries that produce their outputs successively by considering one input element after the other, have to be accepted as genuine stream queries. This intuition on the continuous, successive production of the output can be

formalized by the constraint that the output is determined by finite prefixes of the input. This is the content of the following axiom denoted (FP$^\infty$).

(FP$^\infty$) For all $s \in D^\infty$ and all $u \in D^*$: If $Q(s) \in uD^\infty$, then there is a $w \in D^*$ s.t. $s \in wD^\infty \subseteq Q^{-1}[uD^\infty]$.

As a further aspect of stream processing we consider data-driveness: The actions of the agent that processes the stream (to be read as: the outputs produced by the function Q in this abstract setting) are triggered by the input stream and not by the agent itself—modulo a finite output production before the streaming starts. In particular, this means that finite streams are allowed to be mapped only to finite streams. We state this condition as further axiom:

(F2F) For all $s \in D^*$ it holds that: $Q(s) \in D^*$.

The dual constraint says that infinite streams must be mapped to infinite streams.

(I2I) For all $s \in D^\omega$ it holds that $Q(s) \in D^\omega$.

Already axioms (FP$^\infty$) and (F2F) can be represented by a class of operators based on the notion of abstract computability (Gurevich, Leinders, and Van Den Bussche 2007). The very general notion of an *abstract computable* stream function is that of a function which is incrementally computed by calculations of finite prefixes of the stream w.r.t. a function called *kernel*. Formally: Let $K : D^* \longrightarrow D^*$ be a function from finite words to finite words over D. Then one defines the stream query *Repeat*$(K) : D^\infty \longrightarrow D^\infty$ as *Repeat*$(K) : s \mapsto \bigcirc_{j=0}^{|s|} K(s^{\leq j})$. So, *Repeat*$(\cdot)$ takes the kernel function K and applies it to the growing input stream, consuming one stream element after the other.

Definition 2. *A query Q is* abstract computable *(AC) iff there is a kernel K such that $Q(s) =$ Repeat$(K)(s)$.*

Using a more familiar speak from the stream processing community, the kernel function is a *window function*—and henceforth we refer to K as a window function. But note, that the window content is allowed to grow.

An important aspect for the usability of queries is that queries can be cascaded, or—formally—that they can be composed. This is indeed the case for AC (and also for SAC queries, see below) as shown in (Gurevich, Leinders, and Van Den Bussche 2007).

The definition of abstract computable queries is quite general and, in fact, it does not say anything about the computability of K. The main idea for this definition stems from the theory of computability on real numbers, termed "theory of type-2 effectivity" as outlined in (Weihrauch 2000). The representation theorem mentioned above treads as follows:

Theorem 1 ((Gurevich, Leinders, and Van Den Bussche 2007))**.** *AC queries represent the class of stream queries fulfilling (F2F) and (FP$^\infty$).*

The proof can be found in (Gurevich, Leinders, and Van Den Bussche 2007). We mention here only that for the proof of the direction from (F2F) & (FP$^\infty$) to abstract computability the following window construction is used: $K(\epsilon) = \epsilon$ and $K(sa) = Q(sa) -_\sqsubseteq Q(s)$ for $s \in D^*, a \in D$. With axioms (F2F) and (FP$^\infty$) it can be shown that K is well-defined and gives the right window.

This theorem underlines the importance of the window concept because it shows that any stream query implementing the core idea of an incremental, continuous operation on the ever growing prefix of incoming stream elements can be represented as a window-based query. So it is no surprise that most of the literature on streams has a discussion of window functions in one form or another. Moreover, this representation theorem can be used by the agent engineer to check easily whether a query is AC.

A simple fact following from Theorem 1 is, that if Q maps finite streams to finite streams and is continuous, then for all finite streams s and letters u we have $Q(s) \sqsubseteq Q(su)$, i.e., monotonicity holds. Concretely, we say that a stream query Q is *monotone* iff it fulfills the following axiom:

(MON) For all finite streams $s' \in D^*$ and all (finite and infinite) streams $s \in D^\infty$: If $s' \sqsubseteq s$, then $Q(s') \sqsubseteq Q(s)$.

Proposition 1. *(FP$^\infty$) and (F2F) together entail (MON).*

Theorem 1 corresponds exactly to Theorem 9 of (Gurevich, Leinders, and Van Den Bussche 2007) with the only difference that the authors talk of continuity w.r.t. a topology instead of postulate (FP$^\infty$). Indeed, the sets of the form wD^∞ for $w \in D^*$ can be considered as open balls on top of which open sets and continuity (as usual) can be defined.

The idea of a window as the main core of stream processing on streams has been justified by the characterization in Theorem 1. This still should be true when considering stream queries $Q : D^\omega \longrightarrow D^\omega$ which are defined on infinite streams only and that output infinite streams only. The finite prefix statement then has the following form:

(FP$^\omega$) For all $s \in D^\omega$ and all $u \in D^*$: If $Q(s) \in uD^\omega$, then there is a $w \in D^*$ such that $s \in wD^\omega \subseteq Q^{-1}[uD^\omega]$.

Clearly, any query $Q : D^\omega \longrightarrow D^\omega$ that is generated as *Repeat*(K) for a window function K fulfills (FP$^\omega$). But does the converse hold, too? At least this is not obvious from Theorem 1 because a closer inspection of the proof (which we mentioned directly after the statement of Theorem 1) shows that the constructed window K relies on Q being defined also on *finite streams*. But, fortunately, using a different window construction also yields the desired representation theorem, our first axiomatic contribution in this paper.

Theorem 2. *AC queries of the form $Q : D^\omega \longrightarrow D^\omega$ represent the class of stream queries fulfilling (FP$^\omega$).*

Proof. The proof that AC queries $Q : D^\omega \longrightarrow D^\omega$ fulfill (FP$^\omega$) proceeds exactly in the same way as for functions $Q : D^\infty \longrightarrow D^\infty$ in the proof of Theorem 1.

For the other direction assume that $Q : D^\omega \longrightarrow D^\omega$ fulfills (FP$^\omega$). Let s be any infinite stream $s \in D^\omega$. Consider an ordering of all growing prefixes u_i of $Q(s)$ with $u_0 = \epsilon$ and $u_i = (Q(s))^{\leq i}$. According to (FP$^\omega$) there is, for each u_i, a $w \in D^*$ such that $s \in wD^\omega \subseteq Q^{-1}[u_i D^\omega]$. We may assume that for each u_i we choose its w as the smallest such word w w.r.t. the prefix order. We call the sequence of resulting words $(w_i^s)_{i \in \mathbb{N}}$. Because of the minimality it follows that the sequence is increasing w.r.t. the prefix order. It may be

the case that $w_i^s = w_{i+1}^s$. So we consider the set of indexes $H \subseteq \mathbb{N}$ such that for all $j \in H$ it holds that $w_j \sqsubseteq w_{j+1}$ but $w_j \neq w_{j+1}$, for short: $w_j \sqsubset w_{j+1}$. Assume that H is given as a family $H = (i_k)_{k \in Y}$ where $Y = \mathbb{N}$ or $Y = \{1, \ldots, n\}$ for some natural number n. Now we define the following function K^s for all words $v \sqsubseteq s$. The word v can be of the following form: It is some word where there is a proper growth from w_j to w_{j+1}, i.e., it is a word of the form w_{i_k} or it is a word of the form w_j where no change happens, than it has one of the form $w_{i_k+1}, w_{i_k+2}, \ldots, w_{i_{k+1}-1}$. Or v is not represented as a w_j.

$$K^s(w_j^s) = \begin{cases} u_{i_{k+1}-1} & \text{if} \\ & j \in \{i_0, i_0+1, \ldots, i_{0+1}-1\} \,\&\, k \geq 1 \\ u_{i_{k+1}-1} -_{\sqsubseteq} K(w_{i_{k-1}}^s) & \text{if} \\ & j \in \{i_k, i_k+1, \ldots, i_{k+1}-1\} \,\&\, k \geq 1 \end{cases}$$

$$K^s(w_j^s u) = \epsilon \quad \text{if } u \in D^* \text{ and } w_j^s \sqsubset w_j^s u \sqsubset w_{j+1}$$

Then by definition $\mathit{Repeat}(K^s)(s) = Q(s)$. Of course this is still not the window K we are looking for as K^s still depends on the stream s. But, using some detailed case analysis, which we leave out here, it can be shown that for other streams s' the generated $K^{s'}$ gives the same value for the same word w: $K^s(w) = K^{s'}(w)$. □

4.2 Constant-width Windows

The window characterization in the theorems above allow for unbounded windows K, that is windows that refer to the whole streaming history. The aim of this section is towards characterizing stream queries based on non-growing, i.e., constant-width windows.

We start with a simple, stricter version of the monotonicity axiom, which we call (DISTR) (for distribution). It actually characterizes queries that are completely determined by their outputs for finite streams of length 1.

(DISTR) $\forall s \in D^*$ with $|s| \geq 1$: $Q(s) \in D^*$ and for all $s' \in D^\infty$ with $|s'| \geq 1$: $Q(s \circ s') = Q(s) \circ Q(s')$.

Proposition 2. *Any stream query Q fulfilling (DISTR) fulfills: For all $s \in D^\infty$: $Q(s) = \bigcirc_{i=1}^{|s|} Q(s^{=i})$. In particular such stream queries are AC queries and hence fulfill (FP^∞).*

Proof. We skip the proof here, as it is similar to the proof of Prop. 7 (see below). □

In particular, distributive queries are monotone.

Proposition 3. *(DISTR) entails (MON) and (F2F).*

In order to proof a generalization of Prop. 2 we introduce the general notion of an n-window which corresponds to the notion of a finite window of width n.

Definition 3. *A function $K : D^* \longrightarrow Y$ that is determined by the n-suffixes ($n \in \mathbb{N}$), i.e., that fulfills for all words $w, u \in D^*$ with $|w| = n$ the condition $K(uw) = K(w)$ is called an* n-window. *If additionally $K(s) = \epsilon$, for all s with $|s| < n$, then K is called a* normal n-window. *The set of stream queries generated by an n-window for some $n \in \mathbb{N}$ are called* n-window abstract computable *stream queries, for short* n-WAC *operators. The union* WAC = $\bigcup_{n \in \mathbb{N}}$ *n-WAC is the set of window abstract computable stream queries.*

The following proposition is an immediate consequence of Def. 3.

Proposition 4. *1. 0-windows are constant functions on D^* with $K(w) = K(\epsilon)$ for all $w \in D^*$.*
2. Every i-window with $i \leq j$ is also a j-window.
3. Stream queries fulfilling (DISTR) can be generated by a 1-window. Conversely, if a stream query is generated by a 1-window with $K(\epsilon) = \epsilon$, then it fulfills (DISTR).

A property for stream queries related to (DISTR) is the *filter property*. Roughly, the filter property states that Q filters out elements from the input stream thereby outputting a subsequence. The following axiom gives a possible instantiation of this property:

(FILTER) $Q(\epsilon) = \epsilon$; and for all $s \in D^\infty, u \in D$: $Q(us) = u \circ Q(s)$ or $Q(us) = Q(s)$.

The following proposition follows immediately.

Proposition 5. *All operators fulfilling (FILTER) are AC.*

A stricter version of (FILTER) can be termed *time-invariant filter*, abbreviated by (TI-FILTER). According to this axiom, the decision whether to incorporate the element u into the output stream does not depend on the position of u in the input stream, it is time-invariant.

(TI-FILTER) $Q(\epsilon) = \epsilon$; and for all $s \in D^\infty, u \in D, w \in D^*$: Either $Q(wus) = Q(w) \circ u \circ Q(s)$ or $Q(wus) = Q(w) \circ Q(s)$.

An immediate consequence of the definition is that all time-invariant filter queries are also distributive.

Proposition 6. *(TI-FILTER) entails (DISTR).*

From the formulation of (DISTR), it is a small step towards more specific axioms (FACTOR-n) that, for each $n \in \mathbb{N}$, capture exactly the n-window stream queries.

(FACTOR-n) $\forall s \in D^*$: $Q(s) \in D^*$ and

1. if $|s| < n$, $Q(s) = \epsilon$ and
2. if $|s| = n$, for all $s' \in D^\infty$ with $|s'| \geq 1$: $Q(s \circ s') = Q(s) \circ Q((s \circ s')^{\geq 2})$.

The axiom (DISTR) does not exactly correspond to (FACTOR-1) due to the special case of the empty stream. But the axiom (DISTR-1) defined below does.

Now we can show results corresponding to propositions derived for axiom (DISTR), namely a proposition on the factorization of the query result and the representability by a window-based query.

Proposition 7. *For any $n \in \mathbb{N}$ with $n \geq 1$, a stream query $Q : D^\infty \longrightarrow D^\infty$ fulfilling (FACTOR-n) can be written for $|s| \geq n$ as $Q(s) = \bigcirc_{j=1}^{|s|-n+1} Q(s^{[j,n+j-1]})$.*

Proof. Assume $Q : D^\infty \longrightarrow D^\infty$ fulfills (FACTOR-n). Let $|s| \geq n$ and let $s = r \circ s'$ with $|r| = n$. Then $Q(r \circ s') = Q(r) \circ Q((r \circ s')^{\geq 2})$. On the right factor one can apply again the factorization according to (FACTOR-n). Applying this repeatedly (using induction) gives the desired

representation. (With underlinings we indicate those formulae that are evaluated to give the next equation, resp.)

$$\begin{aligned}
Q(s) &= Q(r \circ s') = Q(r) \circ \underline{Q((r \circ s')^{\geq 2})} \\
&= \underline{Q(r)} \circ Q((r \circ s')^{[2,n+1]}) \circ \underline{Q(((r \circ s')^{\geq 2}))^{\geq 3}} \\
&= \underline{Q(r^{[1,n]})} \circ \underline{Q((r \circ s')^{[2,n+1]})} \circ Q(((r \circ s')^{\geq 3})) \\
&= Q(s^{[1,n]}) \circ Q(s^{[2,n+1]}) \circ \underline{Q(((r \circ s')^{\geq 3}))} \\
&= Q(s^{[1,n]}) \circ Q(s^{[2,n+1]}) \circ Q(s^{[3,n+2]}) \circ \\
&\quad Q(((r \circ s')^{\geq 4})) = \ldots \\
&= Q(s^{[1,n]}) \circ Q(s^{[2,n+1]}) \circ \ldots \circ Q(s^{[|s|-n,|s|]}) \\
&= \bigcirc_{j=1}^{|s|-n+1} Q(s^{[j,n+j-1]})
\end{aligned}$$

For infinite s the assertion follows for that from finite s. □

Proposition 7 can be used to prove the following representation proposition.

Proposition 8. *For any $n \in \mathbb{N}$ with $n \geq 1$, a stream query $Q : D^\infty \longrightarrow D^\infty$ fulfills (FACTOR-n) iff it is induced by a normal n-window K.*

Proof. Let $n \geq 1$. Assume that Q fulfills (FACTOR-n). Define K by $K(w) = \epsilon$ for $|w| < n$. For $w \geq n$ let $K(w) = Q(w^{\geq |w|-n+1})$ = Q-value of n-suffix of w. Then $Q(s) = \bigcirc_{j=0}^{|w|} K(s^{\leq j}) = \epsilon$ for words s with $|s| < n$ by definition of K. For $s \geq n$ one has by Proposition 7

$$\begin{aligned}
Q(s) &= \bigcirc_{j=1}^{|s|-n+1} Q(s^{[j,n+j-1]}) \\
&= \bigcirc_{j=1}^{|s|-n+1} K(s^{\leq n+j-1}) = \epsilon \circ \bigcirc_{j=n}^{|s|} K(s^{\leq j}) \\
&= \bigcirc_{j=1}^{n-1} K(s^{\leq j}) \circ \bigcirc_{j=n}^{|s|} K(s^{\leq j}) \\
&= \bigcirc_{j=0}^{|s|} K(s^{\leq j})
\end{aligned}$$

The other direction (namely, n-window induced functions fulfill (FACTOR-n) and monotonicity) is clear. □

Though this proposition is simple, it is important because it characterizes the input-output behaviour of n-window-based queries by a periodic factorization of the output.

Intuitively, the class of WAC stream queries is a proper class of AC stream queries because the former consider only fixed-size finite portions of the input stream whereas for AC stream queries the whole past of an input stream is allowed to be incorporated for the production of the output stream. A simple example for an AC query that is not a WAC query is the parity query PARITY : $\{0,1\}^\infty \longrightarrow \{0,1\}^\infty$ defined as *Repeat*(K_{par}) where K_{par} is the parity window function $K : \{0,1\}^* \longrightarrow \{0,1\}$ defined as $K_{par}(s) = 1$, if the number of 1s in s is odd and $K_{par}(s) = 0$ else. d The window K_{par} is not very complex, indeed one can show that K_{par} is memory bounded. More concretely, it is easy to find a finite automaton with two states that accepts exactly those words with an odd number of 1s and rejects the others. In other words: parity is incrementally maintainable. But finite windows are "stateless", they cannot memorize the actual parity seen so far. Formally, it is easy to show that any finite-window function is AC^0 computable, i.e., computable by a polynomial number of processors in constant time. On the other hand it is well known by a classical result (Furst, Saxe, and Sipser 1984) that PARITY is not in AC^0.

Axioms (FACTOR-n) are not direct generalizations of the axiom (DISTR) as the $\circ$-factors do not factor disjoint parts of the input stream. A more intuitive set of axioms generalizing (DISTR) are the axioms (DISTR-n) defined as follows:

(DISTR-n) $\forall s \in D^*$: $Q(s) \in D^*$; and if $|s| < n$, then $Q(s) = \epsilon$; and if $|s| = n$, then for all $s' \in D^\infty$ with $|s'| \geq 1$: $Q(s \circ s') = Q(s) \circ Q(s')$.

Operators fulfilling (DISTR-n) give the following factorization representation of the output stream:

Proposition 9. *For any $n \in \mathbb{N}$ with $n \geq 1$, a stream query $Q : D^\infty \longrightarrow D^\infty$ fulfilling (DISTR-n) can be written for $|s| \geq n$ as $Q(s) = \bigcirc_{j=1}^{|s|} Q(s^{[nj-n+1,nj]})$.*

Proof. As above by induction. □

All stream queries fulfilling (DISTR-n) can be characterized by *tumbling n-windows* which are defined as functions $K : D^* \longrightarrow D^*$ such that

$$K(w) = \begin{cases} \epsilon & \text{if } |w| < n \text{ or if } |w| \text{ is not a multiple of } n \\ K(v) & \text{if } |w| \text{ is a multiple with } w = uv \\ & \text{and } |v| = n \end{cases}$$

According to this definition stream queries induced by tumbling windows consider only the last n elements and wait n elements before the content is updated. In words of the streaming community: The window has width n and slide n. According to this definition, in between times i and $i + n$ nothing is outputted to the output stream.

Proposition 10. *For any $n \in \mathbb{N}$ with $n \geq 1$, a stream query $Q : D^\infty \longrightarrow D^\infty$ fulfills (DISTR-n) and $Q(s) = \epsilon$ for all $s \in D^*$ with $|s| < n$ iff it is induced by a tumbling n-window.*

Proof. Follows with Prop. 9. □

The above considerations can be generalized to capture sliding windows with slides greater than 1. Due to space restrictions we leave out the details here.

4.3 Considering Time in the Word Model

A refined notion of abstract computability considered by Gurevich and colleagues (Gurevich, Leinders, and Van Den Bussche 2007) is that of *synchronous abstract computability, SAC* for short, which adds to the condition of abstract computability the condition that K maps the empty stream to the empty stream and that for all other streams the window maps to a finite stream of length 1.

Definition 4. *A query Q is* synchronous abstract computable *(SAC) iff there is a window K with $K(\epsilon) = \epsilon$ and $|K(s)| = 1$ for all $s \neq \epsilon$ such that $Q(s) = Repeat(K)(s)$.*

The authors give an axiomatic characterization of SAC queries (Gurevich, Leinders, and Van Den Bussche 2007, Prop. 18), this time using a property they call "non-predicting". The original definition (Gurevich, Leinders, and

Van Den Bussche 2007, (Def. 17)) says that Q is *non-predicting* iff for all streams s, s' and all $t \in \mathbb{N} \setminus \{0\}$ such that $s^{\leq t} = s'^{\leq t}$ one has $Q(s)^{=t} = Q(s')^{=t}$.

Regarding the reading of the definition it has to be stated explicitly that if s and s' have at least length of t and are the same up to t, then the outcomes $Q(s)$ and $Q(s')$ are also defined up to t and are the same up to t. So, in particular, a non-predicting operator maps infinite streams to infinite streams, i.e., being non-predicting entails (I2I). The authors also assume that being non-predicting entails that finite streams are mapped to finite streams. As we cannot find a plausible reading of non-predictability that entails this fact we rephrase their proposition by explicitly mentioning (F2F).

Proposition 11 (Adaptation of (Gurevich, Leinders, and Van Den Bussche 2007))**.** *SAC queries are the non-predicting queries that fulfill (F2F).*

"Non-prediction" is too weak a notion to capture the property stated under this term. The reason is that also a stream query Q_a that maps every nonempty stream to the finite stream a and the empty stream to itself does not produce streams by looking into the future—and hence should be called non-predicting in an intuitive sense. But surely Q_a is not SAC as it maps infinite streams to a finite stream.

Hence, we consider the following notion of no-dilation computability: A query Q is *no-dilation computable, for short: NDAC* iff it is AC with a window K such that $K(\epsilon) = \epsilon$ and $K(s) \leq 1$ for all $s \in D^\infty$ with $s \neq \epsilon$. Then query Q_a from above is an NDAC query (choosing $K(\epsilon) = \epsilon$ and $K(s) = a$ for $s \neq \epsilon$).

The property corresponding to NDAC is a weakening of non-predictability (and a strengthening of FP^∞):

(FP^∞_{ND}) For all $s \in D^\infty$ and all $u \in D^*$: If $Q(s) \in uD^\infty$, then there is a $w \in D^*$ such that $s \in wD^\infty \subseteq Q^{-1}[uD^\infty]$ **and** $|w| \leq |u|$.

We get the following simple characterization.

Proposition 12. *NDAC queries are exactly the queries fulfilling (FP^∞_{ND}).*

Proof. See window construction in proof of Theorem 2. □

5 Conclusion

As stream processing is an important aspect of the agent paradigm, a formal foundation of streams is of outmost importance for any agent-based model and application. With the general (infinite) word-based framework we considered a sufficiently general, yet simple model of streams which allowed us to specify properties of stream queries and to characterize them. In setting up the stream (reasoning) architecture of an agent, the engineer can rely on these results in order to ensure a specific input-output behaviour when using particular classes of stream queries. Table 1 summarizes the representation results discussed in this paper.

Future work concerns an in-depth axiomatic treatment of memory-bounded stream queries, topological characterizations of SAC and WAC queries as well as modeling multiple stream queries without tag annotations—the overall aim being an axiomatization of belief-state changes in belief-revision style and grounding these in changes of streams of percepts.

Stream query Q	**Axioms**
AC of form	
$Q : D^\infty \longrightarrow D^\infty$	(F2F) & (FP^∞)
$Q : D^\omega \longrightarrow D^\omega$	(FP^ω)
normal n-window	(FACTOR-n)
tumbling n-window	(DISTR-n)
SAC	non-predicting & (F2F)
NDAC	(FP^∞_{ND})

Table 1: Representation Results

References

Arasu, A.; Babcock, B.; Babu, S.; McAlister, J.; and Widom, J. 2004. Characterizing memory requirements for queries over continuous data streams. *ACM Trans. Database Syst.* 29(1):162–194.

Beck, H.; Dao-Tran, M.; and Eiter, T. 2016. Equivalent stream reasoning programs. In Kambhampati, S., ed., *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16)*, 929–935. IJCAI/AAAI Press.

Cowley, A., and Taylor, C. J. 2011. Stream-oriented robotics programming: The design of roshask. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1048–1054.

Furst, M.; Saxe, J. B.; and Sipser, M. 1984. Parity, circuits, and the polynomial-time hierarchy. *Theory of Computing Systems* 17:13–27.

Gurevich, Y.; Leinders, D.; and Van Den Bussche, J. 2007. A theory of stream queries. In *Proceedings of the 11th International Conference on Database Programming Languages*, DBPL'07, 153–168. Berlin, Heidelberg: Springer-Verlag.

Heintz, F.; Kvarnström, J.; and Doherty, P. 2010. Bridging the sense-reasoning gap: Dyknow - stream-based middleware for knowledge processing. *Advanced Engineering Informatics* 24(1):14–26.

Özçep, Özgür. L.; Möller, R.; and Neuenstadt, C. 2014. A stream-temporal query language for ontology based data access. In *KI 2014*, volume 8736 of *LNCS*, 183–194. Springer International Publishing Switzerland.

Perrin, D., and Pin, J. 2004. *Infinite Words: Automata, Semigroups, Logic and Games*. Pure and Applied Mathematics. Elsevier Science.

Rabinovich, A. M. 2003. Automata over continuous time. *Theor. Comput. Sci.* 300(1-3):331–363.

Russell, S. J., and Norvig, P. 1995. *Artificial Intelligence – A Modern Approach*. Prentice Hall.

Tucker, P. A.; Maier, D.; Sheard, T.; and Fegaras, L. 2003. Exploiting punctuation semantics in continuous data streams. *IEEE Trans. on Knowl. and Data Eng.* 15(3):555–568.

Weihrauch, K. 2000. *Computable Analysis: An Introduction*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.