Augmented Integer Linear Recurrences
by Aaron Snook, Manuel Blum (advisor)

***Abstract*—This paper proposes a new sequence type, the Augmented Integer Linear Recurrence. This sequence type finds patterns in sequences with runs of the same integer present. AILRs use linear recurrence inference as a base, and uses invertible sequence transforms to transform sequences into others and recursively infers the post-transformation sequences.**

I. INTRODUCTION

A common technique used in solving mathematical problems is to solve the problem for simple cases and then extrapolate the results to more complex cases using sequence inference. This technique is frequently used by students to solve word problems. However, there are a considerable number of sequences that have a relatively simple pattern that do not fit common definitions of sequences, such as linear recurrences or polynomials. Paths through a two-dimensional Cartesian plane grid often have that the sequence of their x-coordinates or y-coordinates has this property. This paper proposes a new kind of sequence definition, the Augmented Integer Linear Recurrence (AILR) as a solution to this problem. One of the main features of an AILR is the ability to recognize runs of the same integer repeated in a sequence and to see patterns in these runs.

II. MOTIVATION

The problem which motivates this definition is the "ten-second sequence" problem – the fact that not all "easy" sequences fall under the definition of common sequence types. An example of this is the sequence
1 2 2 3 3 3 4 4 4 4 5 5 5 5 5…
Another example of this is the sequence of x-coordinates in the spiral shown in Figure 1, assuming the grid square at the center of the spiral is at x=0:

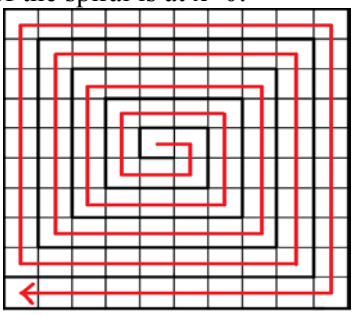

Fig. 1: A spiral.

The sequence of x-coordinates is the following:
0 1 1 0 -1 -1 -1 0 1 2 2 2 2 1 0 -1 -2 -2 -2 -2 -2…

While these sequences have a readily inferable rule, they do not qualify as linear recurrences, polynomials, or other sequences. Therefore, most computer programs fail to recognize these sequences. AILRs are capable of inferring sequences like these, and can solve many simple sequences involving runs of integers and differences. While AILRs and other such sequence inference methods cannot hope to infer all sequences, as doing so would solve several open questions in mathematics and compute uncomputable functions like the Busy Beaver function[1], AILRs can solve many simple sequences that humans would be able to solve but computers would struggle to solve.

III. DEFINITIONS

Linear recurrence: An infinite sequence $S = \{a_i \mid i \in \boldsymbol{N}\}$ of real numbers with the property that there exists a d and a collection of real coefficients $\{k_i | i < d\}$ such that for all $n \geq d$, $a_n = \sum_{i=1}^{d} k_i \, a_{n-i}$. The minimum such d is called the degree of S.

Integer linear recurrence: A linear recurrence in which every term is an integer.

Let S be an integer sequence, whether finite or infinite.

The **difference sequence** of S is the sequence $\{a_{i+1} - a_i | \, i < |S| - 1\}$

Before we define the underlying and mode sequences, we define a partition P(S) $\{A_i | i < f(S)\}$ such that $a_0 \in A_0$and for all $i < |S|, j < f(S)$ if $a_i \in A_j$ and $a_i = a_{i+1}$ then $a_{i+1} \in A_j$ and otherwise $a_{i+1} \in A_{j+1}$. $f(S)$ is the number of sets S needs to be partitioned into this way, and can be infinite if S is infinite.

The **mode sequence** is the sequence of the cardinalities of each A_i in order.

The **underlying sequence** is the sequence of the unique integer in each A_i in order (the integer may appear many times in each A_i).

Note that the difference sequence of a sequence S along with the first term of the original sequence is enough to reconstruct S. Also, the mode and the underlying sequences of a sequence S together contain enough information to construct S.

This allows us to formally define an AILR:
A infinite sequence S is an **AILR** iff:

(1) S is an integer linear recurrence
(2) The difference sequence of S is an AILR
(3) The mode and underlying sequences of S are both AILRs

The **depth** of an AILR S ($depth(S)$) is defined as follows:
The depth of an integer linear recurrence is 0.
The depth of a sequence with an AILR S as its difference sequence is $depth(S) + 1$.
The depth of a sequence with an AILR M as its mode sequence and an AILR U as its underlying sequence is $max\{depth(M), depth(U)\} + 1$.

AILRs $S = \{s_i \mid i \in \boldsymbol{N}\}$ and $T = \{t_i \mid i \in \boldsymbol{N}\}$ are said to be **equivalent** if for all $i \in \boldsymbol{N}, s_i = t_i$.

The **linear recurrence set** of an AILR S ($lrs(S)$) are defined as follows:
If S is an integer linear recurrence, $lrs(S)$ is simply $\{S\}$.
If S has an AILR T as its integer linear recurrence, $lrs(S) = lrs(T)$.
If S has AILRs M and U as its mode and underlying sequences respectively, $lrs(S) = lrs(M) \cup lrs(U)$.

IV. Examples

We use examples from the Online Encyclopedia of Integer Sequences[3] (OEIS).
The difference sequence of a linear recurrence is a linear recurrence (see Appendix, Claim 1). In light of this, (2) in the definition of AILR may seem redundant. However, in conjunction with (3), it allows AILRs to see runs of differences, as shown in the following example.

Consider the sequence (A002260 in the OEIS)
1 1 2 1 2 3 1 2 3 4 1 2 3 4 5…
Its difference sequence is
0 1 -1 1 1 -2 1 1 1 -3 1 1 1 1…
The mode sequence of its difference sequence is
1 1 1 2 1 3 1 4…
The underlying sequence of its difference sequence is
0 1 -1 1 -2 1 -3…
Both the mode and underlying sequences are integer linear recurrences; therefore, they both are AILRs by (1). Therefore by (3) the difference sequence of S is an AILR. Therefore, by (2) the original sequence is an AILR.
Consider the sequence $\{a_i \mid i \in \boldsymbol{N}\}$ where a_i is the ith smallest natural number that is not a square. (A000037 in the OEIS)

Its difference sequence is
1 2 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 …
The mode sequence of its difference sequence is
1 1 3 1 5 1 7…
and the underlying sequence of this difference sequence is
1 2 1 2 1 2 1...
By (1) the mode and underlying sequence of the difference sequence is an AILR, and by (3) that means the difference sequence is an AILR. By (2), the original sequence is an AILR.

V. More Examples

It can be shown that the following are AILRs:

The spiral (x-coordinate) (A174344 in the OEIS):

```
0, 1, 1, 0, -1, -1, -1, 0, 1, 2, 2, 2,
2, 1, 0, -1, -2, -2, -2, -2, -2, -1,
0, 1, 2, 3, 3, 3, 3, 3, 3
```

"A self-generating sequence" (A005041 in the OEIS)

```
1, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6,
6, 7, 7, 7, 7, 8, 8, 8, 8, 9, 9, 9, 9,
10, 10, 10, 10, 10
```

VI. Algorithm

This definition allows for an algorithm to find such sequences. Our algorithm's goal, given a finite sequence of integers, is to obtain a set of continuation functions that are consistent with the sequence of integers. Our algorithm first tests a finite set of integers to see if it is consistent with an integer linear recurrence. If it is, we add that linear recurrence as a continuation function to our final list. We also take the difference, mode, and underlying sequences of the input, and recursively obtain all extension functions consistent with those sequences. If we obtain extensions of the difference sequence, or extensions of the mode and underlying sequences, we construct extension functions of the original sequence using extension functions found in these recursive calls. (See the section "Continuation Functions" for details.) We add all such constructed sequences to the final list

and return. We note that linear recurrence inference is a solved problem[1], and we use an algorithm to do this as part of our algorithm.

VII. Example of Algorithm

Suppose the following finite sequence were input into the algorithm above:
1 1 2 1 2 3 1 2 3 4 1 2 3 4 5
The algorithm would attempt to infer this as an integer linear recurrence and fail. While any sequence of k integers has a linear recurrence interpretation of degree at most $\lfloor\frac{k}{2}\rfloor$, the interpretations may fail to be integer valued; this is the case here.
The algorithm would also attempt to recursively infer the mode and underlying sequences of this sequence, but this will be fruitless in this algorithm, and we will not focus on this.
The algorithm would then obtain the difference sequence D: 0 1 -1 1 1 -2 1 1 1 -3 1 1 1 1
The algorithm would attempt to infer this as an integer linear recurrence and fail.
The algorithm would then determine the mode and underlying sequences of D:
Mode: 1 1 1 2 1 3 1 4
Underlying: 0 1 -1 1 -2 1 -3
which would be inferred as linear recurrences. The mode would be inferred as $L_1 = a_0 = 1, a_1 = 1, a_2 = 1, a_3 = 2,\ a_n = 2a_{n-2} - a_{n-4}$ and the underlying would be inferred as $L_2 = a_0 = 0, a_1 = 1, a_2 = -1, a_3 = 2,\ a_n = 2a_{n-2} - a_{n-4}$. For brevity's sake, we will represent the functions inferred as infinite sequences.
Mode: 1 1 1 2 1 3 1 4 1 5…
Underlying: 0 1 -1 1 -2 1 -3 1 -4…
We then combine these functions to obtain an extension function for D. The mode and underlying functions are combined by for each i starting at 0,
0 1 -1 1 1 -2 1 1 1 -3 1 1 1 1 -4 1 1 1 1 1…
We then use this sequence to construct an extension function of the original sequence:
1 1 2 1 2 3 1 2 3 4 1 2 3 4 5 1 2 3 4 5 6…
which is a logical inference of the original sequence.

VIII. Continuation Functions

In the "Example of Algorithm" section, we represented an AILR as an infinite sequence for simplicity. In reality, we represent an AILR as a recursive function, with linear recurrences as our base case. The AILR inferred in "Example of Algorithm" would be represented as $Difference(Repetition(L_1, L\_2))$, where L_1 and L_2 are as defined in the "Example of Algorithm" section. This means that this AILR was inferred since its difference sequence had mode sequence L_1 and underlying sequence L_2.
To show how this is a continuation function, we need to examine how to extend each kind of AILR by a term.

To extend a sequence of t terms $\{a_1 \ldots a_t\}$ which has been inferred as an integer linear recurrence with degree d and rule $a_n = \sum_{i=1}^{d} k_i a_{n-i}$, obtain the t+1st term using the inferred rule: $a_{t+1} = \sum_{i=1}^{d} k_i a_{t+1-i}$.

To extend a sequence of t terms $\{a_1 \ldots a_t\}$ which has been inferred by inferring its difference sequence S = $\{b_1 \ldots b_{t'}\}$as an AILR, recursively extend S to t terms and obtain a_{t+1} by adding a_t and b_t .

To extend a sequence of t terms $\{a_1 \ldots a_t\}$ which has been inferred by inferring both its mode sequence $M = \{m_1 .. m_{t'}\}$ and its underlying sequence $U = \{u_1 .. u_{t'}\}$, we need a "bank" with two values n and k. Whenever the sequence is extended, if $n > 0$, we add k to the sequence and decrement k. If $n = 0$, we recursively extend M and U by one, and set n to the last term of M and k to the last term of U, and then extend the sequence again.

IX. Runtime

The runtime of the algorithm described is $O(3^m m^3)$ where m is the number of sequence terms inputted and n is the depth limit. To see this, note that the space of sequences checked is a tree with a root of the original sequence. Each node has branching factor 3 (its mode sequence, its underlying sequence, and its difference sequence). Furthermore, note that for any given sequence S, the mode, underlying, and difference sequences have strictly fewer terms than the original sequence (unless the original sequence contains no runs of length greater than one, which can be detected and discarded easily.) At each point in the tree, Gaussian elimination is used to determine the integer linear sequence and whether or not it is integer valued. This algorithm is $O(m^3)$ where m is the number of sequence terms. Note that since we cannot gain terms this bound applies at all points in the tree. This tree can be pruned using several methods (an example would be to not attempt to interpret the mode and underlying sequences of a sequence with no consecutive terms identical). Also, a maximum depth parameter n can be set to the tree to limit its height, trimming the runtime to $O(3^n m^3)$. Having a finite depth parameter is recommended (and the proof of Convergence depends on there being one.)

X. Number of terms used

The more important metric in calculating the efficiency of this algorithm in practice is the number of sequence terms used. For linear recurrences, it takes at least 2d terms to calculate a linear recurrence of degree d[1]. The algorithm implemented does not allow constants to be in the linear recurrence relation. As an example, the form $a_0 = 0,\ a_n = a_{n-1} + 1$ is not allowed. Linear recurrences with a constant in their form can be expressed using ones without such constants, though the degree may increase slightly. For example, the previous linear recurrence can be expressed as $a_0 = 0, a_1 = 1,$ $a_n = 2a_{n-1} - a_{n-2}$, which is of higher degree (but not support). However, AILRs could be implemented with a linear recurrence algorithm that allows constants to be added without changing any features of the program. Furthermore, mode and underlying sequences may be arbitrarily smaller than their originals (the size of the original is the sum of the terms of the mode sequence), preventing any meaningful mathematical guarantees on number of sequence terms required to infer such a sequence. Furthermore, it is possible that several conflicting interpretations will appear for a given finite integer sequence. Eliminating such interpretations may require additional terms (see Convergence).

XI. Conflicts, Integer Constraints

The reason why our definition of AILR includes the requirement that sequence terms be integers is that this filters out several interpretations. Without the integer requirement, any finite sequence of numbers (of length ≥ 2) is an AILR since all such sequences can be interpreted as the start of a linear recurrence[1] In addition, we also filter out interpretations of a mode sequence that include negative integers, since runs of negative length do not exist. The integer requirement does not filter out all undesired interpretations, however, and so multiple differing interpretations can be reached. False interpretations will be eliminated by providing more terms of the sequence to the algorithm once the number of terms given exceeds the number of terms which the false interpretation agrees with the true sequence, but new false interpretations may arise as the number of terms given rises, since this algorithm will infer linear recurrences of degree at most *d/2*, where *d* is the number of sequence terms given. As k rises, the space of integer linear recurrences grows. This fact prevents convergence to a single AILR interpretation. However, with an integral sorting function where AILRs with low degree linear recurrences are ranked higher than ones with ones that contain high-degree sequences, we can ensure that eventually the sequences introduced by adding more terms will all be ranked lower than a single correct interpretation (see Convergence proof).

XII. Ranking Sequences

An issue when inferring sequences in this way is that there may be multiple ways in which a sequence meets the definition of an AILR. For example, the sequence of integers *S* = {1 2 2 3 3 3 4 4 4 4} satisfies the definition of an AILR in the intuitive way: its repetition and mode sequences are each AILRs (the sequences {1 2 3 4}). However, *S* also can be interpreted as an AILR by virtue of being consistent with a degree 5 linear recurrence that happens to match *S* on the first 10 terms, but quickly becomes very negative after that. This sequence is generally considered undesirable. This brings up the issue of sorting the various interpretations by relevance. In general, in sequence inference, we generally consider the simplest (lowest entropy) pattern to be most reliable: lower degree linear recurrences are favored over higher degree ones, lower degree polynomials are favored over higher degree ones, and in general, given multiple sequence interpretations, we would like the one with the simplest Turing machine code to be ranked higher. That is the general principle that we employ in ranking the interpretations found, though there are multiple variables to consider in this case. The major variables are complexity of the linear recurrences involved in the interpretation and the number of transformations used in the interpretation. Both variables are important, and favoring either one over the other has its drawbacks. If we emphasize number of transformations over complexity of linear recurrences, complex linear recurrences will appear and be ranked highly as the number of sequence terms given grows. If we emphasize linear recurrence complexity over number of transformations used, the program will have several inexpensive transformation sequences available to it, allowing it to often find a transformation sequence that produces a simple linear recurrence. The two factors should be given roughly equal weight, but the optimum balance may vary depending on circumstance. Furthermore, ranking functions can be human-aided – whether manually or using databases of common sequences such as the OEIS. One could rank inferred AILRs by for each AILR extending the input finite sequence by a few terms using the AILR's rules, entering the resulting finite sequence into OEIS, and then obtaining the number of results. AILRs with more results would be ranked higher.

A sample ranking function is as follows:

If S is a linear recurrence, *rank(S)* is the degree of the linear recurrence. If *S* has an AILR T as a difference sequence, $rank(S) = rank(T) + 1$. If *S* has an AILR M as its mode sequence and U as its underlying sequence, $rank(S) = max\{rank(S), rank(U)\} + 1$.

XIII. Proofs

Inference

Claim: given an AILR *S*, there exists an *N* such that for all $n \geq N$, if the algorithm is given the first n terms from *S*, *S* will be one sequence inferred.

Proof by structural induction:

Suppose that *S* is an integer linear recurrence of degree *d*. Then, if there are at least *d/2* terms from *S* given, *S* will be inferred by the linear recurrence solver. Suppose that *S* has a difference sequence *T* which is an AILR. By our inductive hypothesis, there exists an *N'* such that for all $n \geq N'$, *T*will be inferred if given n terms. Therefore, if we have at least *N'+1* terms of *S*, we will recursively call the function on the difference sequence with at least *N'* terms, and thus *T* will be inferred and thus *S* will be inferred as well. Suppose that *S* has mode sequence $M = \{m_i \mid i \in N\}$ and underlying sequence *U* which are both AILRs. By our inductive hypothesis, there exist *N', N''* such that for all $n \geq N', M$ can be inferred with *N'* terms, and *U* can be inferred with *N''* terms. Let $N''' = max\{N', N''\}$, and let $N = \sum_{i=0}^{N'''-1} m\_i$. Note that each m_i represents m_i terms in *S*, and so the number of terms represented by the first *N'''* terms is *N*. Therefore, if *S* is given *N* terms, it will recursively call the algorithm with *N'''* terms on *M* and *U*. By the induction hypothesis, *M* and *U* will be inferred in the recursive calls, allowing us to infer *S*.

Convergence

Claim: if the sample rank function in "Ranking Sequences" is used, and a depth limit k is fixed, the following holds: given an AILR S with depth at most k, there exists an N such that for all $n > N$, if the algorithm is given the first n terms from S, an AILR equivalent to S will be the top-ranked function given.

Proof:

Let *N'* be the number of terms required to infer *S*. Let *N''* be the smallest number of terms such that all 3^n transformation sequences of *S* have at least $2 * rank(S)$ terms. Let $N^* = max\{N'', N'\}$. Suppose we give the first N^* terms to our algorithm. By Inference, S will be among the sequences inferred; let $\{F_i \mid i < k\}$ be the list of all interpretations not equivalent to *S* ranked higher than *S* with the input of the first *N* terms. where each F_i is a sequence not equivalent to *S* that is rated higher than *S* and k is the number of such sequences. By definition of non-equivalence to *S*, each F_i disagrees with *S* in the (N_i)th term. Let $N = max\{F_i \mid i < k\}$. Suppose that you give the first *N* terms to the algorithm. Then *S* will still be inferred by Inference, but each F_i will not be inferred since they disagree with the first N^*th terms. Suppose that there is an AILR *T* inferred with *N* terms that is not inferred with N^* terms. This means that there must exist a linear recurrence in *T*'s linear recurrence set that has $degree > rank(S)$, as otherwise *T* would have been inferred with N^* terms as well. $Thus\ rank(T) > rank(S)$. Therefore, this interpretation is ranked below *S*.

Therefore, all interpretations ranked higher than *S* must be equivalent to *S*, and therefore the top ranked AILR will be equivalent to *S* for all $n > N$.

Intuitively, all incorrect sequences simpler than the correct sequence will eventually have a witness to their incorrectness, and therefore once all such sequences have been refuted, the correct sequence will prevail.

Unfortunately, as *N* depends on *S*, which is generally unknown prior to using this algorithm, this proof cannot be used to verify that the algorithm will always produce the correct AILR given a certain fixed number of terms, even if the depth or rank of the AILRs is held below a certain threshold. However, in most practical applications, if the same sequence is ranked on the top for several sufficiently large n, this proof can suggest that you have the right answer.

XIV. Results

When testing against the Online Encyclopedia of Integer Sequences (OEIS), linear recurrences were inferred, as they were in Sam Tetruashvili's work. AILRs that were not linear recurrences were not common, so the inference rate on the OEIS would not be significantly higher for AILRs than for Sam's work. However, there were several interesting sequences on OEIS which were AILRs, not all of which were easy to guess:
The natural numbers in ascending order, except for the perfect squares:
2 3 5 6 7 8 10 11 12 13 14 15 17 18 19…
An oscillating sequence:
0 1 0 -1 0 1 2 1 0 -1 -2 -1 0 1 2 3 2 1 0 -1 -2 -3 -2 -1 0…

XV. Future Work

There is significant room for improvement in conflict resolution strategy for this algorithm, as this is the greatest obstacle from guaranteeing a particular result from this algorithm.

Furthermore, the framework of AILRs suggests a more general sequence-inferring technique. One could propose the following:

A set *F* of algorithms that either infer a sequence or report failure (in the case of AILRs, *F* contained only the integer linear recurrence solve).

A set of transformations *T* (in the case of AILRs, *T* contained the difference, mode, and underlying transformations), partitioned into sets $\{T_1, T_2 \dots T_t\}$ where each T_i is a set of transforms which together can be used to deduce the original sequence.

Then, one could construct a inference function G in the following way:

Let *S* be a given sequence. Attempt to apply all functions in *F* to *S* to find an extension. Otherwise recursively apply all transformations in *T* to *S*. If there is any i such that all transforms in T_i transform *S* into a inferable function by *G*, extend each of the transformed sequences and use them to construct an inference for the original sequence.

In addition, as AILRs are strong with grid related sequences, one could imagine using a generalization of AILRs in a space of higher dimension – the notions of mode, underlying, and difference sequences can be generalized to two-dimensional space. Jerene Yang[2] did work in inference of 2-dimensional sequences, and she used several strategies that were similar to what was done here.

XVI. Appendix

Claim 1: Suppose that *S* is a linear recurrence. Then *S*'s difference sequence is also a linear recurrence.

Proof:

Suppose *S* is an integer linear recurrence of degree *d*. Let the form of *S* be $a_0 = c_0 \dots a_{d-1} = c_{d-1},\ a_n = \sum_{i=1}^{d} k_i a_{n-k}$. Note that the difference sequence of *S* then can be written as $b_0 = c_1 - c_0 \dots b_{d-1} = c_d - c_{d-1},\ b_n = \sum_{i=1}^{d} k_i (a_{n-i+1} - a_{n-i} = \sum_{i=1}^{d} k_i b_{n-i}$.

References

[1] Tetruashvili, Sam and Blum, Manuel. *Inductive Inference of Integer Sequences.* May 9, 2010.

[2] Yang, Jerene, (2012). *Graphical Numerical Inference: AKA Brain Surgery for Excel*. Unpublished manuscript..

[3] OEIS Foundation Inc. (2011), The On-Line Encyclopedia of Integer Sequences, http://oeis.org