ON THE LOGICAL COMPLEXITY OF CYCLIC ARITHMETIC

ANUPAM DAS

ABSTRACT. We study the logical complexity of proofs in cyclic arithmetic (CA), as introduced by Simpson in Sim17, in terms of quantifier alternations of formulae occurring. Writing $C\Sigma_n$ for (the logical consequences of) cyclic proofs containing only $\Sigma_n$ formulae, our main result is that $I\Sigma_{n+1}$ and $C\Sigma_n$ prove the same $\Pi_{n+1}$ theorems, for $n \geq 0$. Furthermore, due to the ‘uniformity’ of our method, we also show that CA and Peano Arithmetic (PA) proofs of the same theorem differ only elementarily in size.

The inclusion $I\Sigma_{n+1} \subseteq C\Sigma_n$ is obtained by proof theoretic techniques, relying on normal forms and structural manipulations of PA proofs. It improves upon the natural result that $I\Sigma_n \subseteq C\Sigma_n$.

The converse inclusion, $C\Sigma_n \subseteq I\Sigma_{n+1}$, is obtained by calibrating the approach of Sim17 with recent results on the reverse mathematics of Büchi’s theorem [KMPM16], and specialising to the case of cyclic proofs.

These results improve upon the bounds on proof complexity and logical complexity implicit in Sim17 and BT17b.

1. Introduction

Cyclic and non-wellfounded proofs have been studied by a number of authors as an alternative to proofs by induction. This includes cyclic systems for fragments of the modal $\mu$-calculus, e.g. NW96 SD03 DHL06 DBHS16 Dou17 AL17, structural proof theory for logics with fixed-points, e.g. San02 FS13 For14 BDS16, (automated) proofs of program termination in separation logic, e.g. BBC08 BDP11 RB17 and, in particular, cyclic systems for first-order logic with inductive definitions, e.g. Bro05 Bro06 BS07 BS11. Due to the somewhat implicit nature of invariants they define, cyclic proof systems can be advantageous for metalogical analysis, for instance offering better algorithms for proof search, e.g. BG12 DP17.

Cyclic proofs may be seen as more intuitively analogous to proofs by ‘infinite descent’ rather than proofs by induction (see, e.g., Sim17); this subtle difference is enough to make inductive invariants rather hard to generate from cyclic proofs. Indeed it was recently shown that simulating cyclic proofs using induction is not possible for some sub-arithmetic languages BT17a, although becomes possible once arithmetic reasoning is available Sim17 BT17b.

Cyclic arithmetic was proposed as a general subject of study by Simpson in Sim17. Working in the language of arithmetic, it replaces induction by non-wellfounded proofs with a certain ‘fairness’ condition on the infinite branches. The advantage of this approach to infinite proof theory as opposed to, say, infinite well-founded proofs via an $\omega$-rule (see, e.g., Sch77), is that it admits a notion of finite proofs: those that have only finitely many distinct subproofs, and so may be represented by a finite (possibly cyclic) graph.

Date: February 19, 2018.
Cyclic arithmetic itself is to cyclic proofs what Peano arithmetic is to traditional proofs. It provides a general framework in which many arguments can be interpreted and/or proved in a uniform manner, and this is one reason why it is arguably a worthy subject of study. This is already clear from, say, the results of [BT17b], where the study of cyclic proofs for pure first-order logic with inductive definitions relied on an underlying arithmetic framework. We make some more comments on this in Sect. 8.

Contribution. In [Sim17], Simpson shows that Peano Arithmetic (PA) (i.e. with induction) is able to simulate cyclic reasoning by proving the soundness of the former in the latter. (The converse result is obtained much more easily.) Nonetheless, several open questions remain from [Sim17], concerning constructivity, normalisation, logical complexity and proof complexity for cyclic and non-wellfounded proofs.

In this work we address the logical complexity and proof complexity of proofs in Cyclic Arithmetic (CA), as compared to PA. Namely, we study how quantifier alternation of proofs in one system compares to that in the other, and furthermore how the size of proofs compare. Writing $\Sigma_n$ for (the logical consequences of) cyclic proofs containing only $\Sigma_n$ formulae, we show, for $n \geq 0$:

1. $\Sigma_{n+1} \subseteq \Sigma_n$ over $\Pi_{n+1}$ theorems (Sect. 4, Thm. 13).
2. CA and PA proofs of the same theorem differ only elementarily in size (Sect. 6, Thm. 26).
3. $\Sigma_n \subseteq \Pi_{n+1}$ over all theorems (Sect. 7, Thm. 28).

1 is obtained by proof theoretic techniques, relying on normal forms and structural manipulations of Peano Arithmetic proofs. It improves upon the natural result that $\Sigma_n \subseteq \Sigma_n$, although induces a non-elementary blowup in the size of proofs. 3 is obtained via a certain ‘uniformisation’ of the approach of [Sim17]. In particular, by specialising the key intermediate results to the case of cyclic proofs, we are able to extract small PA proofs of some required properties of infinite word automata from analogous ones in ‘second-order’ (SO) arithmetic. Finally, 3 is obtained by calibrating the argument of 2 with recent results on the reverse mathematics of Büchi’s theorem [KMPM16], allowing us to bound the logical complexity of proofs in the simulation. Together, these results characterise the logical and proof complexity theoretic strength of cyclic proofs in arithmetic, resolving the questions (ii) and (iii), Sect. 7 of [Sim17].

This paper is structured as follows. In Sects. 2 and 3 we introduce some preliminaries on Peano Arithmetic, proof theory, cyclic proofs and automaton theory. In Sect. 4 we present 1. We introduce some SO theories of arithmetic in Sect. 5 that are conservative over the fragments of PA we need in order to conduct some of the intermediate arguments on infinite word automata. In Sect. 6 we present 2, and in Sect. 7 we adapt the argument to obtain 3. We conclude with some further remarks and perspectives in Sect. 8 including a comparison with the results of [Sim17] and [BT17b].

2. Preliminaries on first-order arithmetic proof theory

We present only brief preliminaries, but the reader is encouraged to consult, e.g., [Bus98] for a more thorough introduction to first-order arithmetic. We work in first-order (FO) logic with equality, with variables written $x, y, z$ etc., terms written $s, t, u$ etc., and formulae written $\varphi, \psi$ etc., construed over $\{\neg, \lor, \land, \exists, \forall\}$. 
We will usually assume formulae are in De Morgan normal form, with negation restricted to atomic formulae. Nonetheless, we may write \( \neg \varphi \) for the De Morgan ‘dual’ of \( \varphi \), defined as follows:

\[
\neg \varphi := \varphi \quad \neg(\varphi \land \psi) := \neg \varphi \lor \neg \psi \quad \neg \forall x. \varphi := \exists x. \neg \varphi
\]

We also write \( \varphi \supset \psi \) for \( \neg \varphi \lor \psi \) and \( \varphi \equiv \psi \) for \( (\varphi \supset \psi) \land (\psi \supset \varphi) \).

Following [Sim17], the language of arithmetic is formulated as \{0, s, +, \times, \lt\}, and a theory is a set \( T \) of closed formulae over this language. We write \( T \vdash \varphi \) if \( \varphi \) is a logical consequence of \( T \). We write \( T_1 \subseteq T_2 \) if \( T_1 \vdash \varphi \) implies \( T_2 \vdash \varphi \), and \( T_1 = T_2 \) if \( T_1 \subseteq T_2 \) and \( T_2 \subseteq T_1 \).

The theory of Robinson arithmetic, written \( Q \), is axiomatised as follows:

\[
i) \forall x, y, z.((x < y \land y < z) \supset x < z).
\]
\[
ii) \forall x, y. (\neg x < y \lor \neg y < x).
\]
\[
iii) \forall x, y.(x < y \lor x = y \lor y < x).
\]
\[
iv) \forall x. \neg x < 0.
\]
\[
v) \forall x, y.(x < y \lor sx < sy).
\]
\[
vi) \forall x. x < sx.
\]
\[
vi) \forall x, y. (\neg x < y \lor \neg y < sx).
\]
\[
viii) \forall x. x + 0 = x.
\]
\[
ix) \forall x, y. x + sy = s(x + y).
\]
\[
x) \forall x. x \cdot 0 = 0.
\]
\[
xi) \forall x. x \cdot sy = xy + x.
\]
\[
\]

Notice that, above and elsewhere, we may write \( \cdot \) instead of \( \times \) in terms, or even omit the symbol altogether, and we assume it binds more strongly than \( + \). We also write \( \forall x \leq t. \varphi \) and \( \exists x \leq t. \varphi \) as abbreviations for \( \forall x.(x \leq t \supset \varphi) \) and \( \exists x.(x \leq t \land \varphi) \) resp. Formulae with only such quantifiers are called bounded.

**Definition 1** (Arithmetical hierarchy). For \( n \geq 0 \), we define:

- \( \Delta_0 = \Pi_0 = \Sigma_0 \) is the class of bounded formulae.
- \( \Sigma_{n+1} \) is the class of formulae of the form \( \exists \vec{x}. \varphi \), where \( \varphi \in \Pi_n \).
- \( \Pi_{n+1} \) is the class of formulae of the form \( \forall \vec{x}. \varphi \), where \( \varphi \in \Sigma_n \).

We say that a sequent is \( \Sigma_n \) (or \( \Pi_n \)) if it contains only \( \Sigma \) (respectively \( \Pi_n \)) formulae.

Notice in particular that, due to De Morgan normal form, if \( \varphi \in \Sigma_n \) then \( \neg \varphi \in \Pi_n \) and vice-versa. In practice we often consider these classes of formulae up to logical equivalence. We say that a formula is in \( \Delta_n \) (in a theory \( T \)) if it is equivalent to both a \( \Sigma_n \) and \( \Pi_n \) formula (provably in \( T \)).

**Definition 2** (Arithmetic). **Peano Arithmetic** (PA) is axiomatised by \( Q \) and the axiom schema of induction:

\[
(1) \quad (\varphi(0) \land \forall x.(\varphi(x) \supset \varphi(sx))) \supset \forall x. \varphi(x)
\]

For a class of formulae \( \Phi \), we write \( \Phi-\text{IND} \) for the set of induction axiom instances when \( \varphi \in \Phi \) in \([1]\). Furthermore, we write \( I\Phi \) for the theory \( Q + \Phi-\text{IND} \).

The following is a classical result:

**Proposition 3** (E.g. [Bus98]). \( \forall \Sigma_n = \Pi_n \), for \( n \geq 0 \).
2.1. A sequent calculus presentation of PA. We will work with a standard sequent calculus presentation of FO logic, as in [Sim17], given in Fig. 1, where \( i \in \{0, 1\} \) and \( a \) is fresh, i.e. does not occur free in the lower sequent. Two important considerations are that we work with cedents as sets, i.e. there is no explicit need for contraction rules, and that we have explicit substitution and equality rules. In the \( \theta\text{-sub} \) rule the ‘substitution’ \( \theta \) is a mapping from variables to terms, which is extended in the natural way to cedents. Substitution is important for the definition of a cyclic arithmetic proof in the next section, but does not change provability in usual proofs. While it can have an effect on proof complexity, this is only up to a polynomial factor once quantifiers are present.

The sequent calculus for \( Q \) is obtained from the FO calculus in the language of arithmetic by adding \( \Delta_0 \) initial sequents for each instantiation of an axiom of \( Q \) by terms. For theories extending \( Q \) by induction axioms, we add the induction rule:

\[
\Gamma \Rightarrow \Delta, \varphi(a) \Rightarrow \varphi(sa), \Delta
\]

Here we require \( a \) to not occur free in the lower sequent. Notice that this satisfies the subformula property, in the ‘wide’ sense of FO logic, i.e. up to substitution of variables by terms. For fragments of PA with induction axioms of bounded logical complexity, we also use the bounded quantifier rules:

\[
\Gamma, a < s, \varphi(a) \Rightarrow \Delta \quad \Gamma, \varphi(a) \Rightarrow \varphi(sa), \Delta
\]

In all cases the eigenvariable \( a \) occurs only as indicated.

The following normalisation result is well-known in the proof theory of arithmetic, and will be one of our main proof-theoretic tools throughout this work:

\begin{figure}
\centering
\begin{align*}
\text{id} & : \Gamma, \varphi \Rightarrow \Delta, \varphi \\
\text{cut} & : \Gamma \Rightarrow \Delta, \varphi \quad \Gamma \Rightarrow \Delta \quad \varphi[a/x] \Rightarrow \Delta \quad \varphi[t/x] \Rightarrow \Delta \\
\text{\theta-sub} & : \Gamma \Rightarrow \Delta \\
\text{cut} & : \Gamma, \varphi \Rightarrow \Delta, \varphi \\
\text{\theta-sub} & : \Gamma \Rightarrow \Delta \\
\text{wk} & : \Gamma, \Gamma' \Rightarrow \Delta, \Delta'
\end{align*}
\end{figure}

\textbf{Theorem 4} (Free-cut elimination, e.g. [Bus98]). Let \( S \) be a sequent system extending FO by the induction rule and some initial sequents closed under substitution. Then any \( S \)-proof can be effectively transformed into one of the same conclusion containing only (substitution instances of) subformulae of the conclusion, an induction formula or a formula in an initial sequent.
Naturally, this applies to the various fragments of $\mathcal{PA}$ that we consider. In particular, notice that a proof in $\Sigma^I_n$ or $\Pi^I_n$ of $\Sigma_n$ or $\Pi_n$ sequents, resp., contains just $\Sigma_n$ or $\Pi_n$ formulae, resp.

A slight issue that will be relevant later in Sect. 4 is that we have not defined $\Sigma_n$ and $\Pi_n$ as being syntactically closed under positive Boolean combinations, even if semantically we know that they are. In fact, this does not cause a problem for the result above, since we can always prenex ‘on the fly’ in a proof by cutting against appropriate derivations. For instance, in a proof, a step of the form,

\[ \Gamma \Rightarrow \Delta, \forall x. \varphi \quad \Gamma \Rightarrow \Delta, \forall y. \psi \]

may be locally replaced by a derivation of the form:

\[
\Gamma \Rightarrow \Delta, \forall x. \varphi \quad \Gamma \Rightarrow \Delta, \forall y. \psi \quad \forall x, y. (\varphi \land \psi) \rightarrow \forall x, y, (\varphi \land \psi)
\]

In a similar way we will often assume that a ‘block’ of existential or universal quantifiers is coded by a single quantifier, using pairings and Gödel $\beta$ functions, whose basic properties are all formalisable already in $\mathsf{I\Delta}_0$ (see, e.g., [Bus98]).

3. PRELIMINARIES ON CYCLIC ARITHMETIC AND AUTOMATA

Before presenting ‘cyclic arithmetic’, we will present the general notion of non-wellfounded proofs in arithmetic, from [Sim17].

**Definition 5.** A **preproof** is a possibly infinite binary tree labeled by sequents in a locally correct manner in the calculus for $\mathcal{Q}$. As in [Sim17], we treat inference steps as nodes of the tree and sequents as edges. A preproof is **regular** if it has only finitely many distinct subtrees or, equivalently, if it is the unfolding of a finite directed graph, possibly with cycles.

The following notions are directly from Dfn. 1 and 2 in [Sim17]:

**Definition 6** (Precursors, traces, $\infty$-proofs). Let $(\Gamma_i \Rightarrow \Delta_i)_{i \geq 0}$ be an infinite branch through a preproof. For terms $t, t'$ we say that $t'$ is a **precursor** of $t$ at $i$ if one of the following holds:

- $\Gamma_i \Rightarrow \Delta_i$ is the conclusion of a $\theta$-sub-step and $t = \theta(t')$.
- $\Gamma_i \Rightarrow \Delta_i$ is the conclusion of an $t_1 = t_2$-step, and $t' = t[t_1/t_2, t_2/t_1]$.
- $\Gamma_i \Rightarrow \Delta_i$ is the conclusion of any other rule and $t = t'$.

A trace along $(\Gamma_i \Rightarrow \Delta_i)_{i \geq 0}$ is a sequence $(t_i)_{i \geq 0}$, for some $n \geq 0$, such that whenever $i \geq n$ the term $t_i$ occurs in $\Gamma_i \Rightarrow \Delta_i$ and,

(a) $t_{i+1}$ is a precursor of $t_i$ at $i$; or
(b) the atomic formula $t_{i+1} < t$ occurs in $\Gamma_{i+1}$, where $t$ is a precursor of $t_i$ at $i$.

When (b) holds, we say that the trace progresses at $i + 1$.

An $\infty$-**proof** is a preproof such that any infinite branch has a trace that progresses infinitely often. If it is regular then we simply call it a **cyclic proof**. $\mathsf{CA}$ is the class of cyclic proofs built over initial sequents corresponding to $\mathcal{Q}$.
The reader may consult [Sim17] for several examples of ∞-proofs. Notably, ∞-proofs are complete for the standard model \( \mathbb{N} \) (Thm. 4, [Sim17]). On the other hand, soundness of ∞-proofs for various logics with respect to a standard model is now a well-known result [Bro00, BST11], but for the current setting it was stated and proved in [Sim17]. We recall the proof since we will have to formalise a variant of it in Sect. 6, and also since the quantifier case in the argument of [Sim17] is omitted, whereas this subtlety will need some consideration when it is formalised.

**Proposition 7** *(Soundness).* If \( \pi \) is an ∞-proof of \( \varphi \), then \( \mathbb{N} \models \varphi \).

*Proof sketch.* Suppose otherwise, i.e. \( \mathbb{N} \models \lnot \varphi \). We will inductively construct an infinite branch \((\Gamma_i \Rightarrow \Delta_i)_{i \geq 0}\) of \( \pi \) and associated assignments \( \rho_i \) of natural numbers to each sequent’s free variables, such that \( \mathbb{N}, \rho_i \not\models \Gamma_i \Rightarrow \Delta_i \). Assuming \( \varphi \) is closed (by taking its universal closure), we set \( \Gamma_0 \Rightarrow \Delta_0 \) to be \( \Rightarrow \varphi \) and \( \rho_0 = \emptyset \).

Each step except for substitution, \( \forall \)-right and \( \exists \)-left constitutes a true implication, so if \( \mathbb{N}, \rho_i \not\models \Gamma_i \Rightarrow \Delta_i \) then \( \rho_i \) also must not satisfy one of its premisses. We may thus choose one such premiss as \( \Gamma_{i+1} \Rightarrow \Delta_{i+1} \) and set \( \rho_{i+1} = \rho_i \).

If \( \Gamma_i \Rightarrow \Delta_i \) concludes a \( \theta \)-sub-step, we may set \( \rho_{i+1} = \theta(\rho_i) \). If \( \Gamma_i \Rightarrow \Delta_i \) concludes a \( \forall \)-right step, let \( \forall x. \varphi \) be the principal formula and assume \( x \) does not occur free in the conclusion. Since \( \mathbb{N}, \rho_i \not\models \Gamma_i \Rightarrow \Delta_i \), we must have that \( \mathbb{N}, \rho_i \not\models \exists x. \lnot \varphi \). We choose a value \( k \in \mathbb{N} \) witnessing this existential and set \( \rho_{i+1} = \rho_i \cup \{ x \mapsto k \} \). The \( \exists \)-left case is dealt with similarly. This infinite branch must have an infinitely progressing trace, say \((t_i)_{i \geq n}\), by the definition of ∞-proof. However notice that, for \( i \geq n \), \( \rho_i(t_i) \geq \rho_{i+1}(t_{i+1}) \) and, furthermore, at a progress point along the trace, \( \rho_i(t_i) > \rho_{i+1}(t_{i+1}) \). Thus, \((\rho_i(t_i))_{i \geq n}\) is a monotone decreasing sequence of natural numbers that does not converge, contradicting the fact that \( \mathbb{N} \) is well-ordered.

Later, in Sect. 6, we will use the fact that the choices for generating an invalid branch in the proof above can be made *uniformly* in an arithmetic setting.

### 3.1. Defining \( C\Sigma_n \)

Simpson proposes in [Sim17] to study systems of cyclic proofs containing only \( \Sigma_n \) formulae, and to compare such systems to \( \Sigma_n \). This is rather pertinent in light of the free-cut elimination result we stated, Thm. 4, any \( C\Sigma_n \)-proof of a \( \Sigma_n \)-sequent can be assumed to contain just \( \Sigma_n \) formulae possibly at a non-elementary cost in proof size, whence the comparison. However, in order to be able to admit routine derivations of more complex formulae, e.g. the \( \Sigma_{n+1} \) law of excluded middle or the universal closure of a \( \Sigma_n \) sequent, we will close this notion under logical consequence.

**Definition 8.** Let \( \Phi \) be closed under subformulae and substitution. \( C\Phi \) is the first-order theory axiomatised by the universal closures of conclusions of cyclic proofs containing only \( \Phi \)-formulae.

Notice that, by the free-cut elimination result, Thm. 4, and the subformula property, any \( C\Sigma_n \) proofs of \( \Sigma_n \)-sequents contain only \( \Sigma_n \)-sequents anyway, without loss of generality. This more ‘robust’ definition allows us to easily compare fragments of cyclic arithmetic. For instance, we have the following:

**Proposition 9.** \( C\Sigma_n = \Sigma n \), for \( n \geq 0 \).

*Proof sketch.* For the left-right inclusion, replace each \( \Sigma_n \) sequent \( \vec{p}, \Gamma \Rightarrow \Delta \) with the sequent \( \vec{p}, \overline{\Delta} \Rightarrow \overline{\Gamma} \), where \( \overline{\Gamma} \) and \( \overline{\Delta} \) contain the De Morgan dual formulae of \( \Gamma \) and \( \Delta \).
and \( \Delta \) resp. and any atomic formulae are amongst \( \vec{p} \). Any traces will be preserved, and the proof can be made correct by adding some extra logical steps. The converse implication is proved in the same way. \( \square \)

Using a standard technique, e.g. from [Bro06], we also can rather simply show the following result:

**Proposition 10.** \( \Sigma_n \subseteq C\Sigma_n \), for \( n \geq 0 \).

Note that, while a similar result was given in [Sim17], that argument rather shows that \( \Sigma_n \subseteq C\Sigma_{n+1} \).

**Proof sketch.** Given an \( \Sigma_n \) proof, we simply simulate every local inference step, the only nontrivial case being an induction step:

\[
\frac{\Gamma \Rightarrow \varphi(0), \Delta \quad \Gamma, \varphi(a) \Rightarrow \varphi(sa), \Delta}{\Gamma \Rightarrow \varphi(t), \Delta}
\]

This is simulated by the following cyclic derivation,

\[
\begin{align*}
\frac{\text{cut}}{\Gamma \Rightarrow \varphi(0), \Delta} & \quad \frac{\text{cut}}{\Gamma \Rightarrow \varphi(b), \Delta} \quad \frac{\text{cut}}{a < b, \Gamma \Rightarrow \varphi(sa), \Delta} \\
\text{sub} & \quad \Gamma, \varphi(a) \Rightarrow \varphi(sa), \Delta & \quad b = sa, \Gamma \Rightarrow \varphi(b), \Delta \\
\text{cut} & \quad 0 = b, \Gamma \Rightarrow \varphi(b), \Delta & \quad 0 < b, \Gamma \Rightarrow \varphi(b), \Delta \\
\text{sub} & \quad \Gamma \Rightarrow \varphi(b), \Delta & \quad \Gamma \Rightarrow \varphi(t), \Delta
\end{align*}
\]

where we have written \( \bullet \) to mark roots of identical subtrees. Any infinite branch that does not have a tail in the proofs of the two premisses of \textit{ind} must hit \( \bullet \) infinitely often. Therefore they admit an infinitely progressing trace alternating between \( a \) and \( b \), with the progress point underlined above. \( \square \)

Notice that, e.g. in the simulation of induction above, traces need not be connected in the graph of ‘ancestry’ of a proof, cf. [Bus98]. This deviates from other settings where it is occurrences that are tracked, rather than terms [DBHS16, BDS16, Dou17].

3.2. B"uchi automata: checking correctness of cyclic proofs. A cyclic pre-proof can be effectively checked for correctness by reduction to the inclusion of ‘Büchi automata’, yielding a \textbf{PSPACE} bound. As far as the author is aware, this is the best known upper bound, although no corresponding lower bound is known. As we will see later in Sect. 6 this is one of the reasons why we cannot hope for a ‘polynomial simulation’ of cyclic proofs in a normal proof system, and so why elementary simulations are more pertinent.

**Definition 11.** A nondeterministic Büchi automaton (NBA) \( A \) is a tuple \( (A, Q, \delta, q_0, F) \) where: \( A \) is a finite set, called the \textbf{alphabet}, \( Q \) is a finite set of \textbf{states}, \( \delta \subseteq (Q \times A) \times Q \) is the \textbf{transition relation}, \( q_0 \in Q \) is the \textbf{initial} state, and \( F \subseteq Q \) is the set of \textbf{final} or \textbf{accepting} states. We say that \( A \) is \textbf{deterministic} (a DBA) if \( \delta \) is a function \( Q \times A \rightarrow Q \). A ‘word’ \( \{a_i\}_{i \geq 0} \in A^\omega \) is \textbf{accepted} or
recognised by \( \mathcal{A} \) if there is a sequence \( (q_i)_{i \geq 0} \in Q^\omega \) such that: for each \( i \geq 0 \), \((q_i, a_i, q_{i+1}) \in \delta \), and for infinitely many \( i \) we have \( q_i \in F \).

From a cyclic preproof \( \pi \) we can easily define two automata, say \( A^\pi_b \) and \( A^\pi_t \) respectively accepting just the branches and just the branches with infinitely progressing traces. See [Sim17] for a construction of \( A^\pi_t \). We point out that \( A^\pi_b \) is essentially just the dependency graph of \( \pi \) with all states final, and so is in fact deterministic. Technically the transition relation here is not total, but this can be ‘completed’ in the usual way by adding a non-final ‘sink’ state for any outstanding transitions. We thus state the now well-known ‘correctness condition’ for cyclic proofs:

**Proposition 12** ([Sim17]). A cyclic preproof \( \pi \) is a \( \infty \)-proof if and only if \( L(A^\pi_b) \subseteq L(A^\pi_t) \).

4. A translation from \( \text{I}_\Sigma_{n+1} \) to \( \text{C}_\Sigma_n \), over \( \Pi_{n+1} \)-theorems

We show in this section our first result, that cyclic proofs containing only \( \Sigma_n \)-formulae are enough to simulate \( \text{I}_\Sigma_{n+1} \):

**Theorem 13.** \( \Sigma_{n+1} \subseteq \text{C}_\Sigma_n \), over \( \Pi_{n+1} \) theorems, for \( n \geq 0 \).

One example of such logical power in cyclic proofs was given in [Sim17], in the form of \( \text{C}_\Sigma_1 \)-proofs of the totality of the Ackermann-Péter function: \( \text{I}_\Sigma_1 \) only proves the totality of the primitive recursive functions [Par72].

To prove the theorem above, we will rather work in \( \text{I}_\Pi_{n+1} \), cf. Prop. 3, since the exposition is more intuitive. We first prove the following intermediate lemma.

**Lemma 14.** Let \( \pi \) be a \( \Pi_{n+1} \) proof, containing only \( \Pi_{n+1} \) formulae, with conclusion

\[
\Gamma, \forall x_1.\varphi_1, \ldots, \forall x_l.\varphi_l \Rightarrow \Delta, \forall y_1.\psi_1, \ldots, \forall y_m.\psi_m
\]

where \( \Gamma, \Delta, \varphi_i, \psi_j \) are \( \Sigma_n \) and \( \vec{x}, \vec{y} \) occur only in \( \varphi, \psi \) respectively. Then there is a \( \text{C}_\Sigma_n \) derivation \( [\pi] \) of the form:

\[
\begin{array}{c}
\Gamma \Rightarrow \Delta, \varphi_i \quad i \leq l \\
\Gamma \Rightarrow \Delta, \psi_1, \ldots, \psi_m
\end{array}
\]

**Proof sketch.** We proceed by induction on the structure of \( \pi \). Notice that we may assume that any \( \Pi_{n+1} \) formulae occurring have just a single outermost \( \forall \) quantifier, by interpreting arguments as pairs and using Gödel’s \( \beta \) functions. This introduces only cuts on formulae of the same form.

We henceforth write \( \vec{\psi} \) for \( \psi_1, \ldots, \psi_m \) and, as an abuse of notation, \( \forall \vec{x}.\vec{\varphi} \) and \( \forall \vec{y}.\vec{\psi} \) for \( \forall x_1.\varphi_1, \ldots, \forall x_l.\varphi_l \) and \( \forall y_1.\psi_1, \ldots, \forall y_m.\psi_m \) respectively. (Notice that this is a reasonable abuse of notation, since the \( \forall \)s can be prenexed outside a conjunction or disjunction already in pure FO logic.) We sketch only the interesting inductive cases here.

Propositional logical steps are easily dealt with, relying on invertibility and cuts, with possible structural steps. Importantly, due to the statement of the lemma,
such steps apply to only $\Sigma_n$ formulae. For instance, if $\pi$ extends a proof $\pi'$ by a $\wedge$-left step,

$$
\begin{align*}
\Gamma, \chi_0, \chi_1, \forall \vec{x}, \vec{\phi} &\Rightarrow \Delta, \forall \vec{y}, \vec{\psi} \\
\Gamma, \chi_0 \land \chi_1, \forall \vec{x}, \vec{\phi} &\Rightarrow \Delta, \forall \vec{y}, \vec{\psi}
\end{align*}
$$

then we define $\lceil \pi \rceil$ as,

$$
\begin{cases}
\chi_0, \chi_1 \Rightarrow \chi_0 \land \chi_1 \\
\Gamma, \chi_0 \land \chi_1 &\Rightarrow \Delta, \phi_i
\end{cases}
$$

$$
\begin{array}{c}
\text{cut} \\
\chi_0, \chi_1 \Rightarrow \chi_0 \land \chi_1 \\
\Gamma, \chi_0 \land \chi_1 &\Rightarrow \Delta, \phi_i
\end{array}
$$

$\lceil \pi' \rceil$

$$
\begin{align*}
\Gamma, \chi_0, \chi_1 &\Rightarrow \Delta, \vec{\psi} \\
\Gamma &\Rightarrow \Delta, \chi_0 \land \chi_1, \vec{\psi}
\end{align*}
$$

and if $\pi$ extends proofs $\pi_0, \pi_1$ by a $\wedge$-right step,

$$
\begin{align*}
\Gamma, \forall \vec{x}, \vec{\phi} &\Rightarrow \Delta, \chi_0, \forall \vec{y}, \vec{\psi} \\
\Gamma, \forall \vec{x}, \vec{\phi} &\Rightarrow \Delta, \chi_1, \forall \vec{y}, \vec{\psi}
\end{align*}
$$

then we define $\lceil \pi \rceil$ as:

$$
\begin{cases}
\Gamma \Rightarrow \Delta, \chi_0 \land \chi_1, \phi_i \\
\chi_0 \land \chi_1 &\Rightarrow \chi_j
\end{cases}
$$

$$
\begin{array}{c}
\text{cut} \\
\Gamma \Rightarrow \Delta, \chi_0 \land \chi_1, \phi_i \\
\chi_0 \land \chi_1 &\Rightarrow \chi_j
\end{array}
$$

$\lceil \pi, \rceil$

$$
\begin{align*}
\Gamma &\Rightarrow \Delta, \chi_j, \vec{\psi} \\
\forall j < 2
\end{align*}
$$

If $\pi$ extends a proof $\pi'$ by a thinning step,

$$
\begin{align*}
\Gamma, \forall \vec{x}, \vec{\phi} &\Rightarrow \Delta, \forall \vec{y}, \vec{\psi} \\
\Gamma', \Pi, \Gamma, \forall \vec{x}, \vec{\phi} &\Rightarrow \Delta, \forall \vec{y}, \vec{\psi}, \Delta', \forall \vec{z}, \vec{\chi}
\end{align*}
$$

where $\Gamma', \Delta', \vec{\chi}$ are $\Sigma_n$ and $\Pi$ is $\Pi_{n+1}$, then we define $\lceil \pi \rceil$ as:

$$
\begin{cases}
\{ \Gamma', \Gamma \Rightarrow \Delta, \phi_i, \Delta' \}_{i \leq l} \\
\Gamma', [\pi'], \Delta'
\end{cases}
$$

$$
\begin{array}{c}
\text{wk} \\
\{ \Gamma', \Gamma \Rightarrow \Delta, \phi_i, \Delta' \}_{i \leq l} \\
\Gamma', [\pi'], \Delta'
\end{array}
$$

$$
\begin{align*}
\Gamma', \Gamma &\Rightarrow \Delta, \vec{\psi}, \Delta' \\
\Gamma', \Gamma &\Rightarrow \Delta, \vec{\psi}, \Delta', \vec{\chi}
\end{align*}
$$

where $\Gamma', [\pi'], \Delta'$ is obtained from $\lceil \pi' \rceil$ by prepending $\Gamma'$ and appending $\Delta'$ to each sequent. For this we might need to rename some variables in $\pi'$ so that eigenvariable conditions are preserved after the transformation. Notice that we are
simply ignoring the extra premisses due to \(\Pi\). If \(\pi\) extends proofs \(\pi_0, \pi_1\) by a cut step on a \(\Pi_{n+1}\) formula,

\[
\begin{array}{c}
\Gamma, \forall z.\varphi \Rightarrow \Delta, \forall y.\psi, \forall z.\chi(z) \\
\Gamma, \forall z.\varphi, \forall z.\chi(z) \Rightarrow \Delta, \forall y.\psi
\end{array}
\]

then we define \([\pi]\) as:

\[
\begin{array}{c}
\{\Gamma \Rightarrow \Delta, \varphi_i\}_{i \leq l} \\
\Gamma \Rightarrow \Delta, \psi, \chi
\end{array}
\]

\[
\begin{array}{c}
\{\Gamma \Rightarrow \Delta, \varphi_i\}_{i \leq l} \\
\{\text{w}k \quad \Gamma \Rightarrow \Delta, \psi, \varphi_i\}_{i \leq l}
\end{array}
\]

\[
\begin{array}{c}
\Gamma \Rightarrow \Delta, \psi, \varphi_i \quad \Gamma \Rightarrow \Delta, \psi
\end{array}
\]

The final dotted 'contraction' step is implicit, since we treat cedents as sets. Again, we might need to rename some variables in \(\pi_1\). If instead the cut formula were \(\Sigma_n\), say \(\chi\), we would define \([\pi]\) as:

\[
\begin{array}{c}
\{\Gamma \Rightarrow \Delta, \varphi_i\}_{i \leq l} \\
\Gamma \Rightarrow \Delta, \chi, \psi
\end{array}
\]

\[
\begin{array}{c}
\Gamma \Rightarrow \Delta, \chi, \psi \quad \Gamma, \chi \Rightarrow \Delta, \psi
\end{array}
\]

If \(\pi\) extends a proof \(\pi'\) by a \(\forall\)-left step,

\[
\begin{array}{c}
\Gamma, \chi(t), \forall z.\varphi \Rightarrow \Delta, \forall y.\psi \\
\Gamma, \forall z.\chi(z), \forall z.\varphi \Rightarrow \Delta, \forall y.\psi
\end{array}
\]

where \(\forall z.\chi(z)\) is \(\Pi_{n+1}\), we define \([\pi]\) as follows:

\[
\begin{array}{c}
\Gamma \Rightarrow \Delta, \chi(c) \\
\Gamma \Rightarrow \Delta, \chi(t)
\end{array}
\]

\[
\begin{array}{c}
\{\text{w}k \quad \Gamma \Rightarrow \Delta, \varphi_i\}_{i \leq l} \\
\Gamma, \chi(t) \Rightarrow \Delta, \varphi_i
\end{array}
\]

\[
\begin{array}{c}
\Gamma \Rightarrow \Delta, \psi \\
\Gamma \Rightarrow \Delta, \chi(t) \Rightarrow \Delta, \psi
\end{array}
\]

If \(\pi\) extends a proof \(\pi'\) by a \(\forall\)-right step,

\[
\begin{array}{c}
\Gamma, \forall z.\varphi \Rightarrow \Delta, \forall y.\psi, \chi \\
\Gamma, \forall z.\varphi \Rightarrow \Delta, \forall y.\psi, \forall z.\chi
\end{array}
\]
where \( \forall z. \chi \) is \( \Pi_{n+1} \), then we define \([\pi]\) as:

\[
\begin{align*}
\{ wk & \quad \Gamma \Rightarrow \Delta, \varphi_i \} \\
& \quad \Gamma \Rightarrow \Delta, \varphi_i, z \\
\end{align*}
\]


\[
[\pi']
\]

\[
\Gamma \Rightarrow \Delta, \vec{\psi}, \chi
\]

Here we may need to again rename some variables in \( \chi \) and apply substitutions at the end. If \( \pi \) extends a proof \( \pi' \) by a \( \exists \)-right step,

\[
\begin{align*}
\Gamma, \forall \vec{x}. \vec{\phi} & \Rightarrow \Delta, \forall \vec{y}. \vec{\psi}, \chi(t) \\
\Gamma, \forall \vec{x}. \vec{\phi} & \Rightarrow \Delta, \forall \vec{y}. \vec{\psi}, \exists z. \chi(z)
\end{align*}
\]

where \( \exists z. \chi(z) \) is \( \Sigma_n \), then we define \([\pi]\) as:

\[
\begin{align*}
\{ wk & \quad \Gamma \Rightarrow \Delta, \varphi_i, \exists z. \chi(z) \} \\
& \quad \Gamma \Rightarrow \Delta, \varphi_i, \chi(t), \exists z. \chi(z) \\
\end{align*}
\]

\[
[\pi'], \exists z. \chi(z)
\]

\[
\exists \Gamma \Rightarrow \Delta, \vec{\psi}, \chi(t), \exists z. \chi(z)
\]

\[
\Gamma \Rightarrow \Delta, \vec{\psi}, \exists z. \chi(z)
\]

Again, some variables in \( \pi' \) might have to be renamed. Any other quantifier steps are dealt with routinely.

Finally, if \( \pi \) extends proofs \( \pi_0, \pi' \) by an induction step,

\[
\begin{align*}
\text{ind} & \quad \Gamma, \forall \vec{x}. \vec{\phi} \Rightarrow \Delta, \forall \vec{y}. \vec{\psi}, \forall z. \chi(c) \Rightarrow \Delta, \forall \vec{y}. \vec{\psi}, \forall z. \chi(sc) \\
\end{align*}
\]

we define \([\pi]\) to be the following cyclic proof,

\[
\begin{align*}
\{ \Gamma \Rightarrow \Delta, \varphi_i \}_{i \leq l} & \quad \Gamma \Rightarrow \Delta, \chi(c) \\
\end{align*}
\]

\[
[\pi']
\]

\[
\Gamma \Rightarrow \Delta, \vec{\psi}, \chi(d)
\]

\[
\Gamma \Rightarrow \Delta, \vec{\psi}, \chi(d)
\]

\[
\Gamma \Rightarrow \Delta, \vec{\psi}, \chi(d)
\]

\[
\Gamma \Rightarrow \Delta, \vec{\psi}, \chi(d)
\]

\[
\Gamma \Rightarrow \Delta, \vec{\psi}, \chi(t)
\]

where we have written \( \bullet \) to mark roots of identical subtrees. Notice that any branch hitting \( \bullet \) infinitely often will have an infinitely progressing trace alternating between \( c \) and \( d \), by the underlined progress point \( c < d \): again, thanks to the eigenvariable conditions, we may rename variables in \([\pi']\) so that no substitutions interfere with this trace. Otherwise an infinite branch has a tail that is already in \([\pi']\) (or \([\pi_0]\)), and so has an infinitely progressing trace by the inductive hypothesis. \( \square \)
The lemma above gives us a simple proof of the main result of this section:

Proof of Thm. 13. Suppose $\pi$ is a $\Pi_{n+1}$ proof of a $\Sigma_n$ sequent, under Prop. 3. By Thm. 4 we may assume that $\pi$ contains only $\Pi_{n+1}$ cuts, whence we may simply apply Lemma 14 to obtain the required result. Notice that, by ‘de-generalising’ the outer $\forall$ of a $\Pi_{n+1}$ theorem, there are only $\Sigma_n$ formulae in the conclusion of $\pi$, possibly with free variables, and so there are no assumption sequents after applying Lemma 14. □

5. SOME SECOND-ORDER THEORIES AND CONSERVATIVITY

Again, we give only brief preliminaries, but the reader is encouraged to consult the standard texts [Sim09, Hir14]. We now consider a two-sorted, or ‘second-order’ (SO), version of FO logic, with variables $X, Y, Z,$ etc. ranging over sets of individuals, and new atomic formulae $t \in X$, sometimes written $X(t)$. We write $Q_2$ for an appropriate extension of $Q$ by basic axioms governing sets (see, e.g., [Sim09] or [Hir14]), and write $\Sigma^0_n$, $\Pi^0_n$ and $\Delta^0_n$ for the classes $\Sigma_n$, $\Pi_n$ and $\Delta_n$ respectively, but now allowing free set variables to occur.

Definition 15. The recursive comprehension axiom schema is the following:

$\text{\Delta}^0_1$-CA : $\forall \bar{y}, \bar{Y}. (\forall x. (\varphi(x, \bar{y}, \bar{Y}) \equiv \neg \psi(x, \bar{y}, \bar{Y})) \supset \exists X. \forall x. (X x \equiv \varphi(x)))$

where $\varphi, \psi$ are in $\Sigma^0_1$ and $X$ does not occur free in $\varphi$ or $\psi$. From here, the theory $\text{RCA}_0$ is defined as $Q_2 + \text{\Delta}^0_1$-CA + $\Sigma^0_1$-IND.

Notice that there is an unfortunate coincidence of the notation CA for ‘comprehension axiom’ and ‘cyclic arithmetic’, but the context of use should always avoid any ambiguity.

Since we will always work in extensions of $\text{RCA}_0$, which prove the totality of primitive recursive functions, we will conservatively add function symbols for primitive recursive functions on individuals whenever we need them. We will also henceforth consider FO theories extended by ‘oracles’, i.e. uninterpreted set/predicate variables. For instance, we write $\Sigma_n(X)$ for the same class of proofs as $\Sigma_n$ but where the symbol $X$ is allowed to occur as a predicate. The following result is an adaptation of well known conservativity results, e.g. as found in [Sim09] [Hir14].

Proposition 16. $\text{RCA}_0 + \Sigma^0_n$-IND is conservative over $\Sigma_n(X)$.

Proof sketch. First we introduce countably many fresh set symbols $X_{\varphi, \psi}^{\bar{t}, \bar{Y}}$, indexed by $\Sigma^0_1$ formulae $\varphi(x, \bar{x}, \bar{X}), \psi(x, \bar{x}, \bar{X})$ with all free variables indicated, FO terms $\bar{t}$ with $|\bar{t}| = |\bar{x}|$ and SO variables $\bar{Y}$ with $|\bar{Y}| = |\bar{X}|$. These will serve as witnesses to the sets defined by comprehension. We replace the comprehension axioms by initial sequents of the form:

$\forall x. (\varphi(x, \bar{t}, \bar{Y}) \equiv \neg \psi(x, \bar{t}, \bar{Y})) \Rightarrow t \in X_{\varphi, \psi}^{\bar{t}, \bar{Y}}$

$\forall x. (\varphi(x, \bar{t}, \bar{Y}) \equiv \neg \psi(x, \bar{t}, \bar{Y})), t \in X_{\varphi, \psi}^{\bar{t}, \bar{Y}} \Rightarrow \psi(t, \bar{t}, \bar{Y})$

It is routine to show that these new initial sequents define a system equipotent to $\text{RCA}_0 + \Sigma^0_n$-IND.
Now we apply free-cut elimination, Thm. 4, to a proof in such a system and replace every positive occurrence of $t \in X(\vec{t}, \vec{Y}, \varphi, \psi)$ with $\varphi(t, \vec{t}, \vec{Y})$, and every negative occurrence, i.e. an occurrence of $t \notin X(\vec{t}, \vec{Y}, \varphi, \psi)$ with $\psi(t, \vec{t}, \vec{Y})$. (Recall here that we assume formulae are in De Morgan normal form.) The comprehension initial sequents are now just identities in the positive case, and purely logical theorems in the negative case. Any extraneous free set variables in induction steps (except $X$), e.g. $Y$, may be safely dealt with by replacing any atomic formula $s \in Y$ with $\top$. The resulting proof is in $I\Sigma_n(X)$. □

It is worth pointing out that the transformation from a SO proof to a FO proof can yield a possibly non-elementary blowup in the size of proofs, due to, e.g., the application of (free-)cut elimination.

5.1. Automaton theory for $\omega$-languages in second-order arithmetic.

From now on we will be rather informal when talking about finite objects, e.g. automata, finite sequences, or even formulae. In particular, we may freely use such objects within object-level formulae, when in fact we are formally referring to their ‘Gödel numbers’. Also, statements inside quotations, “.”, will usually be (provably) recursive in any free variables occurring, i.e. $\Delta^0_1$. This way quantifier complexity is safely measured by just the quantifiers outside quotations.

We may treat a set symbol $X$ as a binary predicate by interpreting its argument as a pair and using Gödel’s ‘$\beta$ functions’ to primitively recursively extract its components. We use such predicates to encode sequences by interpreting $X(x, y)$ as “the $x^{\text{th}}$ symbol of $X$ is $y$”. Axiomatically, this means we presume we have already the totality and determinism of $X$ as a binary relation. Formally, for a set $S$ and a set symbol $X$ treated as a binary predicate, we will write $X \in S^\omega$ for the conjunction of the following two formulae,

\begin{align}
\forall x. \exists y \in S. X(x, y) \\
\forall x, y, z. ((X(x, y) \land X(x, z)) \supset y = z)
\end{align}

i.e. $X$ is, in fact, a function $\mathbb{N} \to S$.

**Definition 17** (Language membership). Let $A = (A, Q, \delta, q_0, F)$ be a NBA and treat $X$ a binary predicate symbol. We define the formula $X \in L(A)$ as:

\begin{align}
X \in A^\omega \land \exists Y \in Q^\omega. & \left( Y(0, q_0) \land \forall x. (Y(x), X(x), Y(x)) \in \delta \\
& \land \forall x, \exists x' > x. Y(x') \in F \right)
\end{align}

If $A$ is deterministic and $X \in A^\omega$, we write $q_X(x)$ for the $x^{\text{th}}$ state of the run of $X$ on $A$. Again, this function is primitive recursive, so its graph is provably total in $\text{RCA}_0$. For DBA, we alternatively define $X \in L(A)$ as:

\begin{equation}
X \in A^\omega \land \forall x. \exists x' > x. q_X(x') \in F
\end{equation}

This ‘double definition’ will not be problematic for us, since $\text{RCA}_0$ can check if an automaton is deterministic or not and, if so, even prove the equivalence between the two definitions:

**Proposition 18.** $\text{RCA}_0 \vdash \forall \text{DBA } A. (\text{(5)} \equiv \text{(6)}).$
Proof sketch. Let $\mathcal{A} = (A, Q, \delta, q_0, F)$ be a deterministic automaton. For the left-right direction let $Y \in Q^\omega$ be an ‘accepting run’ of $X$ on $\mathcal{A}$ and show by induction that $Y(x, q_X(x))$. For the right-left direction, we use comprehension to define an ‘accepting run’ $Y \in Q^\omega$ by:

$$Y(x, q) \equiv q = q_X(x)$$

Clearly the definition of $Y$ is $\Delta^0_1$, and we can show that such $Y$ is a ‘correct run’ by induction on $x$. \hfill \Box

Notice that, for a deterministic automaton, the formula for acceptance is arithmetical in $X$, i.e. there are no SO quantifiers. This will be rather important for uniformity in the simulation of cyclic proofs in the next section.

6. An elementary simulation of CA in PA

In this section we will adapt Simpson’s approach in [Sim17] for showing that $\text{CA} \subseteq \text{PA}$ into a somewhat more uniform result in $\text{PA}$. This essentially constitutes a formalisation of the soundness argument, Prop.[7] in a SO theory conservative over the target fragment of $\text{PA}$. The ‘uniformity’ we aim for ensures that the possibly non-elementary blowup translating from SO proofs to FO proofs occurs once and for all for a single arithmetical theorem. Only then do we instantiate the theorem (inside $\text{PA}$) by the cyclic proof in question, leading to only elementary blowup.

To give an idea of how the result is obtained, we take advantage of the following aspects of the soundness argument for cyclic proofs:

1. The Büchi automaton accepting all infinite branches of a cyclic proof is, in fact, deterministic, and so we can express acceptance of a word in this automaton arithmetically.
2. A branch of invalid sequents and corresponding assignments, as in the proof of Prop.[7] can be uniformly generated from an initial unsatisfying assignment by an arithmetical formula.
3. Morally, the ‘inductive definitions’ we are dealing with have finite closure ordinals, and so we need only arbitrarily often progressing traces, rather than explicit infinitely progressing traces.

Together, these give us just enough ‘wiggle room’ to carry out the soundness argument in a sufficiently uniform way.

Throughout this section we will also carefully track how much quantifier complexity is used in theorem statements, since we will later modify this argument to obtain a converse result to Thm. [13]

6.1. An arithmetically uniform treatment of automata. We define an arithmetical corollary of NBA acceptance that is sufficiently strong to formalise the soundness argument for cyclic proofs:

**Definition 19.** Let $\mathcal{A}$ be a NBA and $X \in A^\omega$. Temporarily let us write:

- $F(x) := “x$ is a finite run of $X$ on $\mathcal{A}$ ending at a final state”.
- $E(z, x, y) := “z$ extends $x$ to a finite run of $X$ on $\mathcal{A}$ hitting $\geq y$ final states”

We define:

$$\text{ArAcc}(X, \mathcal{A}) := \exists x. (F(x) \land \forall y. \exists z. E(z, x, y))$$

Referring to [13] above, we will need the following result:
Proposition 20. \( \text{RCA}_0 \vdash \forall A. (X \in \mathcal{L}(A) \supset \text{ArAcc}(X, A)) \).

For intuition, we may consider \( \omega \)-regular expressions rather than automata, which are of the form \( \sum e_i f_i^2 \), without loss of generality. The formula \text{ArAcc} essentially says that any word accepted by this language has a prefix in some \( e_i \) and tail prefixed by \( f_i^2 \) for each \( n \in \mathbb{N} \).

Proof sketch. Fix \( A = (A, Q, \delta, q_0, F) \) and let \( X \in \mathcal{L}(A) \). By definition of the latter, let \( Y \in Q^\omega \) be an ‘accepting run’ of \( X \) on \( A \), cf. [5]. We have that

\[ (7) \quad \forall y. \exists z \in Q^* \text{ “} z \text{ is a prefix of } Y \text{ hitting } y \text{ final states in } A \text{”} \]

by \((\Sigma^0_1\text{-IND})\) induction on \( y \), appealing to the unboundedness of final states in \( Y \) for both the base case and the inductive steps. Now, in the definition of \text{ArAcc}, we set \( x \) to be the least such \( z \) for which \( (7)[1/y] \) holds. Thus, for any \( y \in \mathbb{N} \), we may find an appropriate \( z \) making \( \text{ArAcc}(X, A) \) true by appealing to \( (7) \). The fact that \( z \) extends \( x \) follows from leastness of \( x \).

Recall that we may freely add symbols for primitive recursive functions to our language. Let us write \( A^c \) and \( A_1 \cup A_2 \) to denote some standard complement and union constructions of automata, say from [KMPM16]. Let \text{Empty}(A) \) be a standard recursive algorithm, expressed as a \( \Delta_1 \) formula in \( A \), determining whether \( A \) computes the empty language (say, also from [KMPM16]). We write “\( A_1 \subseteq A_2 \)” for \text{Empty}((A_1^c \cup A_2)^c). We may now present our main ‘uniform’ result needed to carry out our soundness proof in FO theories:

Theorem 21. \( \text{RCA}_0 + \Sigma^0_2\text{-IND} \) proves:

\[ (8) \quad \forall D_{BA} A_1, \forall NBA A_2. \ (X \in \mathcal{L}(A_1) \land “A_1 \subseteq A_2”) \supset \text{ArAcc}(X, A_2) \]

To prove this, along with Prop. 20 we use the fact that \( \text{RCA}_0 + \Sigma^0_2\text{-IND} \) is strong enough to formalise some fundamental results of automaton theory for \( \omega \)-languages:

Lemma 22 (From [KMPM16]). We have the following:

1. \( \text{RCA}_0 \vdash \forall A. (\text{Empty}(A) \equiv \forall X \in A^\omega. X \notin \mathcal{L}(A)) \). (Lemma 14)
2. \( \text{RCA}_0 \vdash \forall A_1, A_2. (X \in \mathcal{L}(A_1 \cup A_2) \equiv (X \in \mathcal{L}(A_1) \lor X \in \mathcal{L}(A_2))) \). (Thm. 13)
3. \( \text{RCA}_0 + \Sigma^0_2\text{-IND} \vdash \forall A. (X \in \mathcal{L}(A^c) \equiv X \notin \mathcal{L}(A)) \). (Thms. 5 and 12)

Now we can prove the main result of this subsection:

Proof sketch of Thm. 27. Working in \( \text{RCA}_0 \), fix \( A_1, A_2 \), satisfying the antecedent of \( (5) \). We may proceed by Boolean reasoning under Lemma 22 since “\( A_1 \subseteq A_2 \)”.

i.e. \text{Empty}((A_1^c \cup A_2)^c), we have that \( \forall X. X \notin \mathcal{L}((A_1^c \cup A_2)^c) \), by \([1]\). From \( (2) \) we have that \( \forall X. X \in A_1^c \cup A_2 \), whence from \( (2) \) and again \( (2) \) we have that \( \forall X. (X \in \mathcal{L}(A_1) \supset X \in \mathcal{L}(A_2)) \). Finally, since \( X \in \mathcal{L}(A_1) \) from the antecedent, we have \( X \in \mathcal{L}(A_2) \), and so we conclude that \text{ArAcc}(X, A_2) \) by Prop. 20.

By the conservativity result, Prop. 16 we have:

Corollary 23. \( \exists \Sigma_2(X) \) proves \( (8) \).
6.2. Formalising the soundness argument for cyclic proofs. At this point we are able to mostly mimic the formalisation of the soundness argument from [Sim17], although we must further show that a branch of invalid sequents, cf. the proof of Prop. 7, is uniformly describable. Let \( \mathbb{N}, \rho \models_n \varphi \) be an appropriate \( \Sigma_n \) formula in \( \rho, \varphi \) (as long as \( \varphi \) is also \( \Sigma_n \)) asserting that \( \varphi \) is true in \( \mathbb{N} \) under the assignment \( \rho \) of its free variables to natural numbers. This is a standard construction, cf. [Bus98, Kay91].

**Definition 24** (Uniform description of an invalid branch). Let \( \pi \) be a CA-preproof of a sequent \( \Gamma \Rightarrow \Delta \), and let \( n \in \mathbb{N} \) be such that all formulae occurring in \( \pi \) are \( \Sigma_n \). Let \( \rho_0 \) be an assignment such that \( \mathbb{N}, \rho_0 \models_n \Delta \) but \( \mathbb{N}, \rho_0 \not\models_n \Gamma \). The branch of \( \pi \) generated by \( \rho_0 \) is the invalid branch as constructed in the proof of Prop. 7, where at each step when there is a choice of premiss the leftmost one is chosen, and at each step when there is a choice of assignment of a natural number to a free variable the least one is chosen.

We write \( \text{Branch}_n(\pi, \rho_0, x, y) \) for the following predicate:

\[
\text{"the } x^{th} \text{ element of the branch generated by } \rho_0 \text{ in } \pi \text{ is } y"
\]

Notice that \( \text{Branch}_n(\pi, \rho_0, x, y) \) is recursive in \( \models_n \), and so is expressible by a \( \Delta_1(\models_n) \) formula, making it altogether \( \Delta_{n+1} \) in its arguments. In fact, this is demonstrably the case already in \( \text{I} \Sigma_{n+1} \), which can prove that \( \text{Branch}_n(\pi, \rho_0, -, -) \) is the graph of a *function*, as shown in the following proposition. We will write \( \mathcal{A}^{\pi}_b \) and \( \mathcal{A}^{\pi}_f \) for the branch and trace automata of a proof \( \pi \), and \( \text{conc}^{\pi} \) for its conclusion. When we write \( \mathbb{N}, \rho \models_n \Gamma \Rightarrow \Delta \) we mean the \( \Delta_{n+1} \) formula \( (\mathbb{N}, \rho \not\models_n \Delta \lor (\mathbb{N}, \rho \models_n \Gamma)) \).

**Proposition 25.** Let \( \pi \) be a CA-preproof of a sequent \( \Gamma \Rightarrow \Delta \), and let \( n \in \mathbb{N} \) be such that all formulae occurring in \( \pi \) are \( \Sigma_n \). \( \Sigma_{n+1} \) has proofs of size polynomial in \( n \) of:

\[
\forall \pi, \rho_0, (\mathbb{N}, \rho_0 \not\models_n \text{conc}^{\pi}) \supset \text{Branch}_n(\pi, \rho_0, -, -) \in L(\mathcal{A}^{\pi}_b)
\]

**Proof idea.** Using the definition of \( \text{Branch}_n(\pi, \rho_0, -, -) \), the proof follows by a routine induction on the position in the branch. One subtlety is that showing totality requires us to use the *least number principle* for a \( \Sigma_n \)-formula, namely of the form \( \exists x. (\mathbb{N}, \rho \models_n \varphi(x)) \), when dealing with a \( \exists \)-left or \( \forall \)-right step. \( \square \)

On closer inspection, the proof above should go through already in \( \text{I} \Delta_{n+1} \), since we can bound \( y \) in (8), by \( x \) as every state in \( \mathcal{A}^{\pi}_b \) is final. In any case, this does not make a difference to the final bound we obtain.

We are now ready to give our main proof complexity result:

**Theorem 26.** CA = PA, with only elementary difference in proof size.

**Proof sketch.** The right-left direction is available already in [Sim17], or alternatively, by Prop. 9.

For the left-right direction, let \( \pi \) be a CA proof containing only \( \Sigma_n \) formulae. Instantiating \( X \) in Cor. 23 with a \( \Delta_{n+1} \) formula yields a \( \Sigma_{n+2} \) proof of (8), hence from Prop. 25 we arrive at an \( \Sigma_{n+2} \) proof of:

\[
\forall \pi, \rho_0, \left( \supset (\"A^{\pi}_b \subseteq A^{\pi}_f" \land \mathbb{N}, \rho_0 \not\models_n \text{conc}^{\pi}) \right) \supset \text{ArAcc}(\text{Branch}(\pi, \rho_0, -, -), \mathcal{A}^{\pi}_f)
\]

Now, working inside \( \Sigma_{n+2} \) and assuming \( \mathbb{N}, \rho_0 \not\models_n \text{conc}^{\pi} \), we may derive a contradiction from "\( A^{\pi}_b \subseteq A^{\pi}_f \)" by mimicking the proof of Prop. 7. Importantly, if
x witnesses the outer existential of \( \text{ArAcc}(X, \mathcal{A}) \), we must choose a \( y \) bigger than any of the values assigned to terms at the \( x^{th} \) sequent of Branch(\( \pi, \rho_0, --,- \)), i.e. at the fixed position where traces of ‘arbitrary finite progress’ begin. Finally we appeal to the fact that, if \( \pi \) is a correct cyclic proof, there is a proof of \( \text{"A}^n_y \subseteq \text{A}'' \), by exhaustive search. (In fact, such a proof will have exponential size in \( |\pi| \), since inclusion of Büchi automata is decidable in \( \text{PSPACE} \).)

Hence, by the reflection property that \( (N, \emptyset \models_n \varphi) \equiv \varphi \), we have elementary size proofs of \( \text{conc}(\pi) \) in \( \text{I} \Sigma_{n+2} \).  

7. \( \text{I} \Sigma_{n+1} \) CONTAINS \( \text{C} \Sigma_n \)

In fact the proof method we developed in the last section allows us to recover a result on logical complexity too. By tracking precisely all the bounds therein, we obtain that \( \text{C} \Sigma_n \) is contained in \( \text{I} \Sigma_{n+2} \), which is already an improvement to Simpson’s result (see Sect. 8 for a comparison). However, we may actually arrive at the optimal result, given Thm. 13 by more carefully analysing the proof methods of \([\text{KMPM16}]\), namely an inspection of the proofs of Thms. 5 and 12.

**Proposition 27** (Implicit in \([\text{KMPM16}]\)). For any NBA \( \mathcal{A} \), we have that \( \text{RCA}_0 \vdash \forall X \in \text{A}''(X \in \mathcal{L}(\mathcal{A}) \equiv X \notin \mathcal{L}(\mathcal{A})) \).

Notice here that the universal quantification over NBA is external, so that the complementation proofs are not necessarily uniform. This is not a trivial result, since it relies on a version of Ramsey’s theorem, the additive Ramsey theorem, which can be proved by induction on the number of ‘colours’. Usual forms of Ramsey’s theorem are not proved by such an argument, and in fact it is well known that \( \text{RCA}_0 \) cannot even prove Ramsey’s theorem for pairs with only two colours (see, e.g., \([\text{Hir14}]\)).

This allows us to ‘un-uniformise’ the results of the previous section, using Prop. 27 above instead of \( 3 \) from Lemma 22 in order to ‘trade off’ proof complexity for logical complexity.

**Theorem 28.** \( \text{C} \Sigma_n \subseteq \text{I} \Sigma_{n+1} \), for \( n \geq 0 \).

**Proof sketch.** By the definition of \( \text{C} \Sigma_n \), we may assume that all formulae occurring are \( \text{\Sigma}_n \), and hence we may instantiate this value for \( n \) in the proof of Thm. 26. The only difference is that, by using Prop. 27 above instead of \( 3 \) from Lemma 22 in Thm. 21 and Cor. 23, we obtain the analogous results for \( \text{RCA}_0 \) and \( \text{I} \Sigma_1 \) respectively, reducing the induction complexity. Thus the proof of Thm. 26 may be carried out entirely within \( \Sigma_{n+1} \).

7.1. On the proof complexity of \( \text{C} \Sigma_n \). One might be tempted to conclude that the elementary simulation of \( \text{CA} \) by \( \text{PA} \) should go through already for \( \text{C} \Sigma_n \) by \( \text{I} \Sigma_{n+2} \), due to the bounds implicit in the proof of Thm. 26. Furthermore, if we are willing to give up a few more exponentials in complexity, one may even bound the size of \( \text{I} \Sigma_1 \) proofs arising from Prop. 27 by an appropriate elementary function (though this analysis is beyond the scope of this paper).

However, we must be conscious of the ‘robustness’ of the definition of \( \text{C} \Sigma_n \) proofs in terms of complexity. The one we gave, which essentially requires cyclic proofs to contain only \( \text{\Sigma}_n \)-sequents, is more similar to free-cut free \( \text{I} \Sigma_n \) proofs than general ones, so it seems unfair to compare \( \text{I} \Sigma_n \) and \( \text{C} \Sigma_n \) for proof complexity. In fact the
following result allows us to define a more natural notion of a $C_{\Sigma_n}$ proof from the point of view of complexity, while inducing the same theory.

First, let us recall some notions from, e.g., [Bro06, BS11]. Any cyclic proof can be written in ‘cycle normal form’, where any ‘backpointer’ (e.g., the upper sequents we have marked • until now) points only to a sequent that has occurred below it in the proof. Referring to the terminology of [Bro06, BS11] etc., we say that a sequent is a \textit{bud} in a cyclic proof if it has a backpointer pointing to a lower sequent (called the \textit{companion}).

**Proposition 29.** If $\varphi$ has a CA proof whose buds and companions contain only $\Sigma_n$ formulae then $C_{\Sigma_n} \vdash \varphi$.

**Proof idea.** Once again, we apply free-cut elimination, Thm. 4, treating any backpointers as ‘initial sequents’ for a proof in cycle normal form. \hfill $\square$

This again shows the robustness of the definition of $C_{\Sigma_n}$ as a theory, and we would further argue that, from the point of view of proof complexity, the backpointer-condition above constitutes a better notion of $C_{\Sigma_n}$ proof.

At the same time we see that it is not easy to compare the proof complexity of this notion of $C_{\Sigma_n}$ and $I_{\Sigma_{n+1}}$, due to the fact that we have used a free-cut elimination result for the simulations in both directions, inducing a non-elementary blowup in proof size.

**8. Conclusions and further remarks**

In this work we developed the theory of cyclic arithmetic by studying the logical complexity of its proofs. We showed that inductive and cyclic proofs of the same theorems require similar logical complexity, and obtained tight simulation results in both directions. We further showed that the proof complexity of the two frameworks does not differ substantially, although it remains unclear how to properly measure proof complexity for the fragments $C_{\Sigma_n}$, even if the theory seems well-defined and robust. Many of these issues constitute avenues for further work.

**8.1. Comparison to the proofs of [BT17b] and [Sim17].** One reason for our improved quantifier complexity compared to [Sim17], is that Simpson rather relies on \textit{weak König’s lemma} (WKL) to obtain an infinite branch. This, \textit{a priori}, increases quantifier complexity of the argument, since WKL is known to be unprovable in $\text{RCA}_0$ even in the presence of $\Sigma^0_2$-IND; in fact, it is incomparable to $\Sigma^0_2$-IND (see, e.g., [KMPM16]). That said, we point out that the ‘bounded-width WKL’ (bwWKL) of [KMPM16] in fact seems to suffice to carry out Simpson’s proof, and this principle is provable already in $\text{RCA}_0 + \Sigma^0_2$-IND. Applying this strategy to his proof yields only that $C_{\Sigma_n} \subseteq I_{\Sigma_{n+3}}$, since bwWKL is applied to a $\Pi^0_{n+1}$-set, but we believe this can be adapted to a $\Pi^0_{n+2}$ bound, by more carefully tracking quantifier complexity.

We reiterate that the main improvement here is in terms of the proof complexity of the transformation, cf. Thm. 26.

Berardi and Tatsuta’s approach, [BT17b], is rather interesting since it is arguably more ‘structural’ in nature, relying on proof-level manipulations rather than reflection principles. That said there are still crucial sources of nonconstructivity, namely in an ‘arithmetical’ version of \textit{Ramsey’s theorem} (Thm. 5.2) and the consequent \textit{Podelski-Rybalchenko termination theorem} (Thm. 6.1). Both of these increase quantifier complexity by several levels, and so their approach does not seem to yield
comparable logical bounds to this work. However, it does indeed seem from an analysis of their proof that an only elementary blowup occurs when transforming a CA proof to one in PA, matching our Thm. 26.

8.2. Interpreting ordinary inductive definitions in arithmetic. In earlier work by Brotherston and Simpson, cyclic proofs were rather considered over a system of FO logic extended by ‘ordinary’ Martin-Löf inductive definitions [ML71], known as FOL$_{ID}$ [Bro06 BS07 BS11]. Berardi and Tatsuta showed in BT17b that the cyclic system CLKID$^{\omega}$ for FOL$_{ID}$ is equivalent to the inductive system LKID, when at least arithmetic is present, somewhat generalising Simpson’s result [Sim17]. We point out that the two results are arguably equipotent since ordinary Martin-Löf inductive definitions can be interpreted in arithmetic, with the necessary properties provable. This is because the closure ordinals for ordinary Martin-Löf inductive definitions are $\leq \omega$, and so a $\Sigma_1$ inductive construction of ‘approximants’ can always determine whether an individual belongs to an inductive predicate or not. Notice that this was crucial for our use of ArAcc over the SO acceptance formula. This is also precisely the role of the ‘stage numbers’ in BT17b; there the fresh inductive predicates $P'$ can be expressed as $\Delta_0$-formulæ. In particular, this means that CLKID$^{\omega}$(+PA) is conservative over CA. We reiterate that the interest behind the results of [BT17b] is rather the structural nature of the transformations, but this observation also exemplifies why CA is a natural and canonical object of study, cf. Sim17.

8.3. Cyclic propositional proof complexity. One perspective gained from this work comes in the setting of propositional proof complexity (see, e.g., CN10 Kra95). From the results and methods herein, we may formalise in C$\Delta_0(X)$, say, a proof of the relativised version of the (finitary) pigeonhole principle, which is known to be unprovable in I$\Delta_0(X)$ due to lower bounds on propositional proofs of bounded depth [KPW95 PB93].

At the same time the ‘Paris-Wilkie’ translation [PWS1], which fundamentally links I$\Delta_0$ to bounded-depth proofs, works locally on a proof, at the level of formulæ. Consequently one may still apply the translation to the lines of a CA$\Delta_0$ proof to obtain small ‘proof-like’ objects containing only formulæ of bounded depth, and a cyclic proof structure. One would expect that this corresponds to some strong form of ‘extension’, since it is known that adding usual extension to bounded systems already yields full ‘extended Frege’ proofs. However at the same time, some of this power has been devolved to the proof structure rather than simply at the level of the formulæ, and so could yield insights into how to prove simulations between fragments of Hilbert-Frege systems with extension.

References


