A complete axiomatization of MSO on infinite trees

Anupam Das Colin Riba
Laboratoire LIP, ENS de Lyon, CNRS, Inria, UCBL, Université de Lyon

***Abstract*—We show that an adaptation of Peano's axioms for second-order arithmetic to the language of MSO completely axiomatizes the theory over infinite trees. This continues a line of work begun by Büchi and Siefkes with axiomatizations of MSO over various classes of linear orders.**

Our proof formalizes, in the axiomatic theory, a translation of MSO formulas to alternating parity tree automata. The main ingredient is the formalized proof of positional determinacy for the corresponding parity games, which as usual allows to complement automata and to deal with negation of MSO formulas. The Comprehension Scheme of monadic second-order logic is used to obtain *uniform* winning strategies, where most usual proofs of positional determinacy rely on instances of the Axiom of Choice or of transfinite induction.

I. Introduction

Rabin's tree theorem [10], that the monadic theory of two successors (S2S) is decidable is one of the most powerful results concerning decidability of logics. In this paper, we present a complete axiomatization of S2S, i.e. a set of sound MSO-formulae from which one can infer all of S2S. Our axiomatization can be seen as a subsystem of second order Peano's arithmetic (PA2).

We continue a line of work begun by Büchi and Siefkes, who gave axiomatizations of MSO on various classes of linear orders. (see e.g. [12], [2]). These works essentially rely on formalization of automata in the logic. A major result in the axiomatic treatment of logics over infinite structures is Walukiewicz's proof of completeness of Kozen's axiomatization of the modal μ-calculus [17]. More recent related works include complete axiomatizations of MSO and of the modal μ-calculus over finite trees [14], [6]. These works (as well as the recent reworking [11] of Siefkes's completeness proof of MSO on ω-words [12]), rely on model-theoretic techniques, which allow elegant reformulations of algebraic approaches. For infinite trees, the algebraic approach is much more difficult [1]. We therefore directly formalize a translation of formulas to automata in the axiomatic theory.

There are several ways to translate MSO formulas to automata (see e.g. [7], [15], [18]). We choose to translate formulas to alternating parity automata. The two non-trivial steps in the translation of formulas to alternating automata are negation and projection (existential quantification). For negation, one needs to complement automata, while for projection, one has to simulate an alternating automata by an equivalent non-deterministic one (non-determinization).

Recall that alternating tree automata generalize non-deterministic automata by allowing positive boolean formulas in transitions. We adopt (a special case of) the presentation of [18], in which transitions are represented using a double powerset (corresponding to irredundant disjunctive normal forms), which allows for a smooth treatment of complementation and non-determinization.

Working in the language of MSO imposes some constraints on the formalization. This is mainly due to the fact that it is not possible to compare the length of tree positions, since this would allow to define a Choice formula on tree positions contradicting [8] (see also [3]). As a consequence, when formalizing acceptance of tree automata, we can only deal with positional strategies. General strategies are represented by trees of arbitrary finite branching. Such trees can be representable in MSO, but the relation between a play in an acceptance game and its underlying position in the input tree is not MSO definable, while positional strategies can be described using memoryless plays for which this relation is definable in MSO.

It is well-known (see e.g. [5], [9], [18]) that non-determinization of alternating automata is strongly linked to McNaughton's Theorem (determinization of ω-word automaton). For this, we rely on already known axiomatizations of MSO on ω-words [12], [11]. They provide, for each formula of MSO on trees defining an ω-regular language, a (provably equivalent) formula describing a deterministic ω-word automaton recognizing this language.

The main difficulty concerns complementation. As usual, we rely on positional determinacy of parity games. We show that our axiomatization of MSO proves that acceptance games of alternating automata are positionally determined. Our proof formalizes a variant of the positional determinacy argument of [15]. The main difficulty is the obtention of *uniform* winning strategies: from the description of a set of positions from which (say) player Eloise has a winning strategy, to compute a strategy for Eloise which is winning from all positions of the set. This property is usually naturally proved the Axiom of Choice or some form of well-founded induction. We show that (together with induction on tree positions) the Comprehension axiom scheme allows to build such a strategy.

It is well known that SkS, the monadic theory of k-ary trees, can be embedded in S2S [10]. However, it seems not direct to use such an embedding to obtain axiomatizations of SkS from an axiomatization of S2S. We therefore axiomatize the MSO theory of the full infinite D-ary tree, for D a arbitrary finite ordered set. We denote by MSO_D the corresponding formal systems.

As expected, formalization in (subsystems of) arithmetic

Work partially funded by the ANR Blanc project RÉCRÉ.

involves a lot of coding. But since it is a decidable logic, MSO on infinite trees does not allow the usual primitive recursive codings (see e.g. [13]). We therefore use boolean codings of finite structures. We work in an extension of MSO_D that we call *bounded* FSO_D, where FSO_D stands for *functional second-order* logic. This is basically an extension of MSO_D with function variables and a choice axiom. In *bounded* FSO_D, functions are required to range over finite sets only, so that *bounded* FSO is directly interpretable in MSO_D. Working in bounded FSO_D allows to smoothly handle much of the bureaucracy caused by the finite boolean codings required by the formalization.

We also at some places use *Henkin models*. Thanks to the corresponding completeness result, they allow to reason in genuine mathematical structures rather than at the syntactic level of formal proofs. This eases the presentation of some arguments.

A potentially interesting aspect of this work is that it may open the way to new algorithms, based on proof search, to decide MSO formulas on infinite trees. This seems quite remarkable since besides the algebraic approach of [1], up to now the only approaches to Rabin's theorem proceed *via* translation of formula to automata. We elaborate on this in Sect. VII.

Full proofs can be found in the full version of the paper [4].

Organization of the paper

We present our axiomatization in Sect. II. Section III then gathers the logical tools we need, in particular: basic facts about MSO_D and its relation to MSO on ω-words, as well as the logic FSO_D and its relation to MSO_D. The formalization of alternating parity tree automata is then presented in Sect. IV, which also contains (assuming positional determinacy of parity games) proofs of correctness of basic operations on automata (complementation, projection etc). The proof of completeness of our axiomatization, based on the formalized translation of formulas to automata, is then assembled in Sect. V. Section VI contains our main technical contribution, namely the proof of positional determinacy. We elaborate on possible further directions in Sect. VII and give concluding remarks in Sect. VIII.

II. An axiomatization of MSO on infinite trees

For the whole paper we assume given countable sets $\mathcal{V}^\iota = \{x, y, z, \dots\}$ and $\mathcal{V}^o = \{X, Y, Z, \dots\}$ of resp. *individual* and *monadic* variables. We also fix a non-empty set D of the form $[n] := \{0, \dots, n-1\}$ for some $n \in \mathbb{N}$.

A. The logic MSO_D

Definition II.1 (Language of MSO_D)**.** *The individual terms of MSO_D, denoted a, b, etc. are build from individual variables and the symbols $\dot\varepsilon$ (nullary, intended to denote the root of the tree) and S_d for each $d \in D$ (all unary, intended to denote immediate successors).*

The formulas, denoted $\phi, \psi \in \Lambda_D$, are build from atomic formulas of the form Xa by means of negation $\neg\phi$, disjunction $\phi \vee \psi$ and existential quantifications $\exists x\, \phi$ and $\exists X \phi$.

The connectives $\wedge, \to, \forall x, \forall X$ are defined as usual.

TABLE II
Axioms for MSO_D

Tree Axioms: $\forall x \neg(\mathsf{S}_d(x) \doteq \mathsf{S}_{d'}(x))$ (for each $d \neq d' \in D$)
For each $d \in D$:

$$\forall xy(\mathsf{S}_d(x) \doteq \mathsf{S}_d(y) \to x \doteq y) \qquad \forall x \neg(\mathsf{S}_d(x) \doteq \dot\varepsilon)$$

Induction:

$$\forall X \left[X\dot\varepsilon \to \forall y \bigwedge_{d \in D} (Xy \to X\mathsf{S}_d(y)) \to \forall y Xy \right]$$

Comprehension: for all formula ϕ,

$$\exists X \forall y\, [Xy \longleftrightarrow \phi] \qquad (X \notin \mathrm{FV}(\phi))$$

1) Axiomatization: Provability in MSO_D is obtained by deduction in predicate calculus (rules of Table I, where Γ denotes a finite unordered list of formulas), using the axioms of Table II. We write $\Gamma \vdash_{\mathsf{MSO}_D} \phi$ if $\Gamma \vdash \phi$ is derivable in MSO_D and $\mathsf{MSO}_D \vdash \phi$ for $\vdash_{\mathsf{MSO}_D} \phi$.

2) Interpretation: MSO_D is interpreted in the standard model $\mathfrak{T}_D$ of the D-ary tree D^* as expected: individual variables $x \in \mathcal{V}^\iota$ range over tree positions D^*, $\dot\varepsilon$ is interpreted by the root $\varepsilon \in D^*$, successors S_d map $a \in D^*$ to $a \cdot d \in D^+$ and monadic variables $X \in \mathcal{V}^o$ range over $\mathcal{P}(D^*)$.

Our axiomatization is sound, in the sense that

Fact II.2. *If $\mathsf{MSO}_D \vdash \phi$ then $\mathfrak{T}_D \models \phi$.*

B. Main result

The main result of this paper is that the axiomatization MSO_D is complete w.r.t. MSO on the infinite D-ary tree. This is an immediate consequence (*via* the soundness of MSO_D w.r.t. the full D-ary tree) of the following result, which is our main contribution:

Theorem II.3 (Completeness of MSO_D)**.** *For every closed formula ϕ of MSO_D, either $\mathsf{MSO}_D \vdash \phi$ or $\mathsf{MSO}_D \vdash \neg\phi$.*

Corollary II.4. *For every closed formula ϕ of MSO_D, $\mathsf{MSO}_D \vdash \phi$ if and only if $\mathfrak{T}_D \models \phi$.*

The argument for Thm. II.3 can be outlined as follows: We first formally prove in MSO_D that every formula is equivalent to a formula representing a tree automaton. For a closed formula, the corresponding automaton works on the singleton alphabet. We show that for the formula (denoted $\mathcal{L}(\mathcal{A})$) representing an automaton $\mathcal{A}$ on the singleton alphabet, we have either $\mathsf{MSO}_D \vdash \mathcal{L}(\mathcal{A})$ or $\mathsf{MSO}_D \vdash \neg\mathcal{L}(\mathcal{A})$.

III. A logical toolbox

In this section we introduce the various tools and definitions which we use to formalize the completeness argument and proofs in later sections.

A. Henkin completeness

We recall here the notions of Henkin models and the corresponding completeness results we use. They allow to reason in genuine mathematical structures rather than at the

TABLE I
DEDUCTION RULES

Rules for Propositional Logic

$$\frac{}{\Gamma \vdash \phi \vee \neg\phi} \qquad \frac{}{\Gamma, \phi \vdash \phi} \qquad \frac{\Gamma \vdash \phi \qquad \Gamma \vdash \neg\phi}{\Gamma \vdash \psi}$$

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \phi \vee \psi} \qquad \frac{\Gamma \vdash \psi}{\Gamma \vdash \phi \vee \psi} \qquad \frac{\Gamma \vdash \phi \vee \psi \qquad \Gamma, \phi \vdash \varphi \qquad \Gamma, \psi \vdash \varphi}{\Gamma \vdash \varphi}$$

Rules for First-Order Logic
(where either $\mathcal{X} \in \mathcal{V}^\iota$ and $\mathcal{Y}$ is a term, or $\mathcal{X}, \mathcal{Y} \in \mathcal{V}^o$)

$$\frac{\Gamma \vdash \phi[\mathcal{Y}/\mathcal{X}]}{\Gamma \vdash \exists \mathcal{X} \phi}$$

$$\frac{\Gamma \vdash \exists \mathcal{X} \phi \qquad \Gamma, \phi \vdash \psi}{\Gamma \vdash \psi} \ (\mathcal{X} \notin \mathrm{FV}(\Gamma, \psi))$$

syntactic level of formal proofs. This eases the presentation of some arguments.

An *Henkin structure* $\mathfrak{M}$ for monadic second-order order logic is given by a set $\mathfrak{M}^\iota$ of individual and a set $\mathfrak{M}^o \subseteq \mathcal{P}(\mathfrak{M}^\iota)$ of monadic predicates (over which range resp. the individual variables x, y, z etc., and the monadic variables X, Y, Z etc.)

An *Henkin structure* for MSO_D is an Henkin structure $\mathfrak{M}$ equipped with a constant $\varepsilon^{\mathfrak{M}} \in \mathfrak{M}^\iota$ and functions $\mathsf{S}_d^{\mathfrak{M}} : \mathfrak{M}^\iota \to \mathfrak{M}^\iota$ for each $d \in D$. Such a structure is an *Henkin model* of MSO_D if it satisfies all the axioms of MSO_D, and if moreover $\mathfrak{M} \models \mathfrak{a} \doteq \mathfrak{b}$ iff $\mathfrak{a} = \mathfrak{b}$ for all $\mathfrak{a}, \mathfrak{b} \in \mathfrak{M}^\iota$.

We have as usual (see e.g. [16]):

Theorem III.1 (Henkin Completeness)**.** *Given a closed formula ϕ of MSO_D, we have $\mathsf{MSO}_D \vdash \phi$ iff $\mathfrak{M} \models \phi$ for all Henkin model $\mathfrak{M}$ of MSO_D.*

B. Some basic facts on MSO_D

Equality is definable by its usual second-order coding $(x \doteq y) := \forall X(Xx \to Xy)$. MSO_D proves the usual axioms $\forall x(x \doteq x)$ and $\forall X \forall xy\,[x \doteq y \to Xx \to Xy]$.

The prefix order is definable as

$$(x \dot{\leq} y) \quad := \quad \forall X\,[\mathrm{Up}(X) \to Xx \to Xy]$$

where $\mathrm{Up}(X) := \bigwedge_{d \in D} \forall x\,[Xx \to X(\mathsf{S}_d(x))]$. The strict prefix order is $(x \dot{<} y) := (x \dot{\leq} y) \wedge \neg(x \doteq y)$.

The following are important usual facts.

Proposition III.2. *The following are provable in MSO_D:*

(i) $\dot{\leq}$ is reflexive, transitive, antisymmetric, and form a tree domain with root $\dot{\varepsilon}$:

$$\forall x(\dot{\varepsilon} \dot{\leq} x) \quad \forall xyz[y \dot{\leq} x \to z \dot{\leq} x \to (y \dot{\leq} z \vee z \dot{\leq} y)]$$

(ii) $\dot{<}$ is antireflexive, symmetric and unbounded.

(iii) $\dot{<}$-induction:

$$\forall X\,[\forall x\,(\forall y(y \dot{<} x \to Xy) \to Xx) \to \forall x Xx]$$

(iv) Every non $\dot{<}$-minimal element has a predecessor:

$$\forall x\,[\exists y(y \dot{<} x) \to \exists y\,(y \dot{<} x \wedge \neg\exists z(y \dot{<} z \dot{<} x))]$$

An important and useful consequence of $\dot{<}$-induction is the

Proposition III.3 (Recursion Theorem)**.** *Let $\phi(X, x)$ be a formula such that MSO_D proves*

$$\forall x \forall XY[\forall y \dot{<} x(Xy \longleftrightarrow Yy) \to (\phi(X, x) \longleftrightarrow \phi(Y, x))]$$

Then $\mathsf{MSO}_D \vdash \exists! X \forall x[Xx \longleftrightarrow \phi(X, x)]$

Infinite paths are definable as

$$\mathrm{Path}^\infty(X) := \begin{array}{l} \mathrm{Ub}(X) \wedge \mathrm{Lin}(X) \wedge \\ \forall xyz\,\left[Xx \to Xy \to x \dot{\leq} z \dot{\leq} y \to Xz\right] \end{array} \tag{1}$$

where $\mathrm{Ub}(X) := \forall x\,[Xx \to \exists y(Xy \wedge x \dot{<} y)]$
and $\mathrm{Lin}(X) := \forall xy\,\left[Xx \to Xy \to (x \dot{\leq} y \vee y \dot{\leq} x)\right]$.

One of the most important properties of infinite paths in MSO_D is Prop. III.5 below.

C. MSO on ω-words

For $D = [1] = \{0\}$, the standard model $\mathfrak{T}_D$ is the standard model $(\mathbb{N}, \mathcal{P}(\mathbb{N}), (n \mapsto n+1))$ of ω-words, and MSO_D completely axiomatizes its MSO theory [12]. We shall use a stronger fact, namely that for any finite D, relativization to any (provably) infinite path of MSO_D gives the MSO theory of ω-words.

Definition III.4 (Relativization)**.** *Given two formulas ϕ and $\psi(y)$ with disjoint free variables, the relativization of ϕ to $\psi(y)$, denoted $\phi^{\psi(y)}$ (or simply ϕ^ψ when y is clear from the context), is obtained by recursively replacing in ϕ every first-order quantification $\exists x \varphi$ by $\exists x(\psi(x) \wedge \varphi^{\psi(y)})$.*

Proposition III.5. *Consider a closed MSO formula ϕ build on the purely relational language consisting of the relation $(\_ \dot{<} \_)$ only. Then $\mathrm{Path}^\infty(X) \vdash_{\mathsf{MSO}_D} \phi^{X(-)}$ if and only if $(\mathbb{N}, \mathcal{P}(\mathbb{N}), <) \models \phi$.*

MSO formula build as in Prop. III.5 are the same as those considered in the axiomatization of MSO on ω-words of [11].

D. Functional second-order logic

In this section we present an extension of MSO_D with unary function symbols, denoted FSO_D, and called 'functional second-order' (FSO) logic. It is in this system that we will conduct the proofs in later sections.

In particular we show that, when the codomains of these functions are provably finite, with explicit bounds, we have only a conservative extension of MSO_D.

We work in this larger theory mainly for presentational reasons, but also to ease reasoning inside a model of second-order logic, which will now give us access to functions as well as individuals and sets.

Definition III.6 (The system FSO_D)**.** *Assume that* $\mathbf{2} \subseteq D$*. The language of* FSO_D *is that of* MSO_D *together with countably many* unary *function symbols* f, g *etc., possibly with subscripts. We allow quantification over these symbols, i.e. if* ϕ *is a formula then* $\exists f \phi$ *is a formula, and extend the notion of 'free variable' appropriately.*

Terms*, denoted* a, b *etc., are build from individual variables, the nullary symbol* $\dot{\varepsilon}$*, the unary symbols* S_d*, and using unary function variables:* $f(a)$ *is a term if* a *is a term. A term is* closed *if no free variables occur in it.*

The rules of FSO_D *extend* MSO_D *as follows:*

- *In the '*$\exists$*' rules in Table I, the symbols* $\mathcal{X}, \mathcal{Y}$ *are further allowed to range over function symbols.*
- *We add the* Choice *axiom schema,*

$$\forall x \exists y \phi \;\to\; \exists f \forall x\, \phi[f(x)/y] \qquad (\text{for } f \notin \mathrm{FV}(\phi)) \quad (2)$$

Henkin completeness (Thm. III.1) extends to FSO_D w.r.t. the following notion of model: An *Henkin structure* for FSO_D is an Henkin structure $\mathfrak{M}$ for MSO_D equipped with an additional set of functions $\mathfrak{M}^{\iota \to \iota} \subseteq \mathfrak{M}^{\iota} \to \mathfrak{M}^{\iota}$, over which range the function variables f, g etc. Such a structure is a *model* of FSO_D if it is a model of MSO_D and satisfies the additional axioms of FSO_D.

Our main use for Choice is to define new functions in FSO_D. In these cases, it is often implicit that we might be using the following consequence of it, a version of comprehension for functions.

Definition III.7 (Functional Comprehension)**.** *We define the* Functional Comprehension *schema, where* $f \notin \mathrm{FV}(\phi)$*, as*

$$\forall x \exists! y \phi \;\to\; \exists f \forall x y\, (f(x) \doteq y \longleftrightarrow \phi) \quad (3)$$

Proposition III.8. $\mathsf{FSO}_D \vdash (2) \to (3)$.

Proof. We reason inside an arbitrary Henkin model of FSO_D. By the antecedent of (3), we have that $\forall x.\exists y.\phi$, whence we can apply Choice to obtain,

$$\exists f. \forall x. \phi[f(x)/y] \quad (4)$$

which implies one direction of the equivalence in the succedent of (3).

Now let $\mathfrak{f}$ be the function witnessing (4) above. We have that $\phi[\mathfrak{f}(x)/y]$, but again by the antecedent of (3) we have that any witness for $\forall x.\exists y.\phi$ is unique, and so equal to $\mathfrak{f}(x)$, giving the other direction of the equivalence required. □

In the definition of boundedness below, it may seem that we use an odd bounding relation (the newly defined '$\dot{\trianglelefteq}$'). In fact it does not matter which relation we use, as long as it is well-founded, but the current presentation allows us to present the interpretation of FSO_D in MSO_D, Def. III.11 without the need for cumbersome codings.

Definition III.9 (Boundedness)**.** *For words* $\sigma, \tau \in \mathbf{2}^*$*, let us write* $\sigma \trianglelefteq \tau$ *if* $|\sigma| = |\tau|$ *and* σ *is lexicographically less than or equal to* τ*.*

In FSO_D *we define the following:*

$$x \mathrel{\dot{\trianglelefteq}} \overline{\sigma} \quad := \quad \bigvee_{\tau \trianglelefteq \sigma} x = \overline{\tau}$$

where $\sigma, \tau \in \mathbf{2}^*$ *and* $\overline{\sigma}$*,* $\overline{\tau}$ *are the corresponding terms build using* $\dot{\varepsilon}$*,* S_0 *and* S_1*.*

We define 'bounded' functional quantifiers as follows, where b *is a closed term build using* $\dot{\varepsilon}$*,* S_0 *and* S_1*:*

$$\begin{array}{lll} \exists f \mathrel{\dot{\trianglelefteq}} b.\phi & : & \exists f.(\forall x. f(x) \mathrel{\dot{\trianglelefteq}} b) \wedge \phi \\ \forall f \mathrel{\dot{\trianglelefteq}} b.\phi & : & \forall f.(\forall x. f(x) \mathrel{\dot{\trianglelefteq}} b) \to \phi \end{array}$$

A formula is bounded *if it is logically equivalent to one where each function quantifier is bounded by a* closed *term.*

An FSO_D*-proof is* bounded *if every formula occurring in it is bounded.*

Notice, in particular, that an instance of Functional Comprehension is bounded if its Comprehension formula (ϕ in (3) above) is bounded.

Our aim in this section is to obtain the following result:

Theorem III.10. *Bounded* FSO_D *is interpretable in* MSO_D*.*

It might serve as helpful intuition to consider 0 and 1 as defined truth constants standing for falsum and truth,[1] respectively. We can then associate vectors of formulae with bit strings specifying their respective truth values, as below.

Definition III.11 (Interpretation of FSO_D in MSO_D)**.** *For a tuple* $\vec{X} = (X_1, \ldots, X_k)$ *and a bit string* $\sigma \in \mathbf{2}^k$ *we use the following notation:*

$$\vec{X} x \longleftrightarrow \sigma \quad := \quad \bigwedge_{i=1}^{k} X_i x \longleftrightarrow \sigma_i$$

The interpretation $\langle \cdot \rangle$ *of* FSO_D*-formulae commutes with the connectives* $\neg, \vee$ *and* $\exists$ *over individual and set variables. We define* $\langle \exists f \mathrel{\dot{\trianglelefteq}} \sigma.\phi \rangle$ *as,*

$$\exists X_1, \ldots, X_{|\sigma|}. \langle \phi \rangle_{f,\sigma}$$

where $\langle \phi \rangle_{f,\sigma}$ *is obtained from* $\langle \phi \rangle$ *by replacing all atomic subformulae of the form* $X f(t)$ *with:*

$$\bigvee_{\tau \trianglelefteq \sigma} X\overline{\tau} \wedge (\vec{X} a \longleftrightarrow \tau)$$

In order to show that this interpretation is correct, i.e. preserves validity, it will be helpful for us to establish a partial converse to Prop. III.8.

Proposition III.12. *If* ϕ *is bounded, then* $\mathsf{FSO}_D \vdash (3) \to (2)$*.*

Proof. We work in an arbitrary Henkin model of FSO_D. Notice that any bounded instance of Choice has antecedent of the form $\forall x.\exists y \mathrel{\dot{\trianglelefteq}} \overline{\sigma}.\phi(x, y)$. In particular we can find the

[1] For example by defining 0 as $(\dot{\varepsilon} \doteq \mathsf{S}_0(\dot{\varepsilon}))$ and 1 as $\neg 0$.

least $\tau \trianglelefteq \sigma$ such that $\phi(x, \overline{\tau})$ by propositional logic, i.e. we have:

$$\forall x.\exists!y.\phi(x,y) \wedge (\forall z \phi(x,z).y \dot{\trianglelefteq} z)$$

Now we can apply (bounded) Functional Comprehension to obtain,

$$\exists f \dot{\trianglelefteq} \overline{\sigma}.\forall x, y.f(x) = y \iff (\phi(x,y) \wedge (\forall z \phi(x,z).y \leq z))$$

whence we can deduce by first-order logic,

$$\exists f \dot{\trianglelefteq} \overline{\sigma}.\forall x.\phi(x, f(x))$$

which is equivalent to the succedent required. □

Theorem III.13. *The interpretation $\langle\cdot\rangle$ is correct, i.e. if bounded-$\mathsf{FSO}_D \vdash \phi$ then $\mathsf{MSO}_D \vdash \langle\phi\rangle$.*

Corollary III.14. *Bounded FSO_D is a conservative extension of MSO_D.*

For the remainder of this work, when we say 'by Comprehension', we may mean either Comprehension or Functional Comprehension.

E. Some remarks on codings of finite objects

We will fix a coding $\ulcorner\cdot\urcorner$ from finite structures to $\mathbf{2}^*$. We do not give much detail on this coding, in fact it is rather unimportant since we only use its properties internal to defined formulae, so it would not be a problem for the coding itself to vary between definitions.

We assume that all finite objects we use are enumerated and that $\mathbf{2}^k$ contains codes of the first $|\mathbf{2}|^k$ objects. Notice this means that there are infinitely many codes for the same object: one of each sufficiently large length. In this case, within any formula, $\ulcorner\cdot\urcorner$ should map each object to its code of smallest length that also admits a code for every other object occurring in the formula.

This is purely due to the notion of boundedness we use above (via the relation $\trianglelefteq$), and is not a critical feature.

For an n-ary relation $\mathcal{R}$ and objects $a_1, \ldots, a_n$ let $\ddot{\mathcal{R}}\ (\overline{\ulcorner a_1 \urcorner}, \ldots, \overline{\ulcorner a_n \urcorner})$ be the truth value of $\mathcal{R}(a_1, \ldots, a_n)$. Similarly, given closed terms $t_1, \ldots, t_n$, let $\ddot{\mathcal{R}}\ (t_1, \ldots, t_n)$ denote the following boolean formula over $\doteq, \dot{\in}, \mathsf{S}_0, \mathsf{S}_1, t_1, \ldots, t_n$ computing $\mathcal{R}$:

$$\bigvee_{\overline{a} \in \mathcal{R}} \bigwedge_{i=1}^{n} t_i \doteq \overline{\ulcorner a_i \urcorner}$$

We similarly use this 'double dot' notation for infix and other relation symbols when speaking about the codes of objects.

Since the syntax $\overline{\ulcorner\cdot\urcorner}$ is quite heavy, we will systematically admit a certain abuse of this notation: we associate every finite object we use with the closed term computing its code, e.g. we associate $[n] = \{0, \ldots, n-1\}$ with $\overline{\ulcorner [n] \urcorner}$.

In light also of the aforementioned comments on bounding in proofs, we give some examples of the sort of expressions we will work with.

For a finite set A, with a ranging over its elements, we may write the following:[2]

$$\begin{aligned} x \ddot{=} a \quad &: \quad x \doteq \overline{\ulcorner a \urcorner} \\ \forall x \ddot{\in} A\ .\ \phi \quad &: \quad \forall x. \bigwedge_{a \in A} x \doteq \overline{\ulcorner a \urcorner} \to \phi[\overline{\ulcorner a \urcorner}/x] \\ \exists X \ddot{\subseteq} A.\phi \quad &: \quad \exists X. \forall x \dot{\in} X. \bigvee_{B \subseteq A} x \ddot{\in} \overline{\ulcorner B \urcorner} \end{aligned}$$

We may also write,

$$f \text{ to } A \quad : \quad \forall x\ .\ \bigvee_{a \in A} f(x) \ddot{=} \overline{\ulcorner a \urcorner}$$

and so, for example, the following expression,

$$\exists f \text{ to } B^A\ .\ \forall y, z \ddot{\in} A\ .\ f(x)(y) \ddot{=} f(x)(z) \to y \ddot{=} z$$

is read, using our abuse of notation, as,

$$\exists f. \bigvee_{F \in B^A} f(x) \ddot{=} F \wedge \bigwedge_{a, a' \in A} F(a) \ddot{=} F(a') \to a \ddot{=} a'$$

stating that there is a function f that, on input x, outputs the code of an injection from A to B.[3]

All of these expressions are considered bounded, due to equivalence under first-order logic.

Notice also that the use of the double dot notation should prevent ambiguity in such circumstances, although we retain the code notation in the case that it eases parsing of a formula.

IV. Games, automata and closure under logical operations

The focus of this section is to define automata accepting infinite tree languages, define their acceptance using abstract games, and use this setting to show that the ω-regular tree languages are closed under basic logical operations.

A. Abstract games

In this section we introduce concepts and notation to deal with abstract games over a product of the infinite tree D^*.

Recall that we may abuse notation by using meta-notation for a finite set in place of the term computing its code.

Let us fix disjoint finite sets P and O of *proponent* (or "Eloise") and *opponent* (or "Abelard") *labels*, respectively. In fact these will be parameters at the meta-level, and this becomes relevant in Sect. IV-D4.

Definition IV.1 (Game positions)**.** *A* game position *is a pair (x, a) such that,*

$$a \ddot{\in} \mathsf{P} \sqcup \mathsf{O} \tag{5}$$

and a set of game positions is a pair (X, f) such that:

$$\forall x \dot{\in} X\ .\ f(x) \ddot{\subseteq} \mathsf{P} \sqcup \mathsf{O} \tag{6}$$

[2] In these examples we give a reading up to equivalence in first-order logic, to aid presentation.

[3] It also states that, on any input, the output is a function from A to B.

We also reserve variables u, v, w etc. to vary over game positions and U, V, W etc. to vary over sets of game positions, and use the following notation,

$$\begin{array}{rcl}(x,a) \mathrel{\dot=} (y,b) & := & x \mathrel{\dot=} y \wedge a \mathrel{\ddot=} b \\ (x,a) \mathrel{\dot\in} (X,f) & := & x \mathrel{\dot\in} X \wedge a \mathrel{\ddot\in} f(x) \\ (X,f) \mathrel{\dot\subseteq} (Y,g) & := & X \mathrel{\dot\subseteq} Y \wedge \forall x \mathrel{\dot\in} X . f(x) \mathrel{\ddot\subseteq} g(x) \\ \exists v.\phi & := & \exists x . \exists y \mathrel{\dot\in} \mathsf{P} \sqcup \mathsf{O} \ . \ \phi[(x,y)/v] \\ \exists V.\phi & := & \exists X . \exists f \text{ to } \mathcal{P}(\mathsf{P} \sqcup \mathsf{O}) \ . \ \phi[(X,f)/V]\end{array}$$

where, in the $\exists$ cases, we choose x, X, f not free in ϕ.

We define f_{P} and f_{O} by Comprehension, denoting the P *and* O *parts of the range of f, i.e.*

$$\begin{array}{rcl} f_{\mathsf{P}}(x) & \doteq & f(x) \mathrel{\ddot\cap} \mathsf{P} \\ f_{\mathsf{O}}(x) & \doteq & f(x) \mathrel{\ddot\cap} \mathsf{O}\end{array}$$

We assume other set-theoretic and quantifier notations for game positions are defined accordingly from the above.

Recall that D is a finite set of tree-directions.

Definition IV.2 (Games and plays)**.** *We define* games*, or* edge functions*, as follows:*

$$\mathrm{Game}(e, \mathsf{P}, \mathsf{O}) := \quad \forall x \ . \ e(x) \mathrel{\ddot\subseteq} (\mathsf{P} \times \mathsf{O}) \sqcup (D \times \mathsf{O} \times \mathsf{P})$$

We define e_{P} and e_{O} by Comprehension, denoting the parts of e consisting of just P*-edges and just* O*-edges respectively. I.e.*

$$\begin{array}{rcl} e_{\mathsf{P}}(x) & \doteq & e(x) \mathrel{\ddot\cap} \mathcal{P}(\mathsf{P} \times \mathsf{O}) \\ e_{\mathsf{O}}(x) & \doteq & e(x) \mathrel{\ddot\cap} \mathcal{P}(D \times \mathsf{O} \times \mathsf{P})\end{array}$$

We use the following notation,

$$\begin{array}{rcl}(x,a) \xrightarrow[e_{\mathsf{P}}]{} (y,b) & := & x \mathrel{\dot=} y \wedge (a,b) \mathrel{\ddot\in} e_{\mathsf{P}}(x) \\ (x,a) \xrightarrow[e_{\mathsf{O}}]{} (y,b) & := & \bigvee_{d \in D} \mathsf{S}_d(x) \mathrel{\dot=} y \wedge (d,a,b) \mathrel{\ddot\in} e_{\mathsf{O}}(x)\end{array}$$

and: $u \xrightarrow[e]{} v := u \xrightarrow[e_0]{} v \ \vee \ u \xrightarrow[e_1]{} v$.

For this arrow notation, we further borrow some conventions from rewriting theory:

$$V : u \xrightarrow[e]{\omega} \infty \ : \ u \mathrel{\dot\in} V \wedge \forall v \in V . \exists! w \mathrel{\dot\in} V . v \xrightarrow[e]{} w$$

$$u \xrightarrow[e]{*} v \ : \ \exists V : u \xrightarrow[e]{\omega} \infty \ . \ v \mathrel{\dot\in} V$$

We define a play *V from v of a game e, as follows:*

$$\begin{array}{rcl} & & \mathrm{Game}(e, \mathsf{P}, \mathsf{O}) \\ \mathrm{Play}(V, u, e) & : & \wedge \ \ V : u \xrightarrow[e]{\omega} \infty \\ & & \wedge \ \ \forall v \mathrel{\dot\in} V \ . \ u \xrightarrow[e]{*} v\end{array}$$

Notice that, from the point of view of general game graphs, the above definition is a little nonstandard, due to the peculiarities of the setting that we are in:

1) For general game graphs the definition does not discount the possibility that V contains a cycle, or otherwise non-wellfounded set, disconnected from the intended play. This is not a problem in our tree setting.
2) Plays are thus required to be infinite, so if a game is not deadlock-free then it is possible it has no plays.

B. Automata, acceptance and winning

We use a notion of *alternating* automaton, which is essentially a special case of that used in [18]. We typically consider *parity* acceptance conditions, also known as 'Rabin chain' acceptance, but present automata with arbitrary conditions below.

Informally, an alternating automaton is to an automaton what an alternating Turing machine is to a Turing machine: the transition function may code arbitrary boolean conditions on the state and letter read.

In what follows, we fix a finite alphabet Σ.

Definition IV.3 (Alternating tree automata)**.** *An* alternating tree automaton *is a tuple:*

$$\mathcal{A} \quad = \quad (Q, q^{\imath}, \delta, \Omega) \tag{7}$$

where Q is a finite set of 'states', $q^{\imath} \in Q$ is the 'initial' state, $\Omega \subseteq Q^{\omega}$ is the acceptance condition and δ is the 'transition function', having the following format:

$$\delta : Q \times \Sigma \to \mathcal{P}(\mathcal{P}(Q \times D))$$

To define what it means for an automaton to accept a tree, we refer to the notions of game introduced earlier.

Let us fix an automaton $\mathcal{A}$, as specified in (7). In what follows we will specialize the concepts of Sect. IV-A by setting $\mathsf{P} = Q$ and $\mathsf{O} = \mathcal{P}(Q \times D)$.

We use the function variable t as a parameter coding some Σ-labelled tree, i.e. we intend that $\forall x . t(x) \mathrel{\ddot\in} \Sigma$.

Definition IV.4 (Acceptance games)**.** *We define the* acceptance game *$e^{\mathcal{A},t}$ of $\mathcal{A}$ on t by Comprehension as follows:*[4]

$$\begin{array}{rcl} e^{\mathcal{A},t}_{\mathsf{P}}(x) & \ddot= & \{(q,\gamma) \mid \gamma \mathrel{\ddot\in} \delta(q, t(x))\} \\ e^{\mathcal{A},t}_{\mathsf{O}}(x) & \ddot= & \{(d,\gamma,q) \mid (q,d) \mathrel{\ddot\in} \gamma\} \\ e^{\mathcal{A},t}(x) & \doteq & e^{\mathcal{A},t}_{\mathsf{P}}(x) \mathrel{\ddot\sqcup} e^{\mathcal{A},t}_{\mathsf{O}}(x)\end{array}$$

Let $c : \mathsf{P} \to [0, n]$ and (X, f) be a set of game positions.[5]

We define the formula $\mathrm{Par}^c(X, f)$ *as follows:*

$$\begin{array}{rcl} & & \mathrm{Path}^{\infty}(X) \wedge \forall x . \exists! y . y \mathrel{\ddot\in} f(x) \\ \wedge & \bigvee_{i=0}^{\lfloor \frac{n}{2} \rfloor} & \left[\begin{array}{rl} & \forall x \in X . \exists y \in X \ . \ y \geq x \wedge c(f_{\mathsf{P}}(y)) \mathrel{\ddot=} 2i \\ \wedge & \exists x \in X . \forall y \in X \ . \ y \geq x \to c(f_{\mathsf{P}}(y)) \mathrel{\ddot\geq} 2i \end{array} \right]\end{array}$$

where $\mathrm{Path}^{\infty}(X)$ *states that X is an infinite path along D^*, as specified in* (1)*.*

C. Strategies

Acceptance for an automaton will be defined by the existence of a winning P-strategy under the parity condition above. Games with such conditions are known to be *positionally determined*: there is a winning strategy for P or O that is

[4]Technically, in this definition, the occurrence of δ (as well as that of c in the definition of Par^c below) should be double-dotted, since it is a relation over the code represented by the term $t(x)$. However we choose to abuse notation once more in favor of keeping the syntax light.

[5]Below we actually abuse notation by associating a singleton $\{q\}$ with its only element q. This could be rectified by defining c instead as a function $\{\{q\} : q \in \mathsf{P}\} \to [0, n]$.

history-free, i.e. one whose moves are dependent only on the current position and not the history of the play. This choice of condition is ultimately why we are able to express acceptance in bounded FSO_D (and so MSO_D).

Recall that in bounded FSO_D, we can not define a relation compare the length of tree positions, since this would allow to define a Choice formula on tree positions, thus contradicting [8] (see also [3]).

As a consequence, when formalizing acceptance of tree automata, we can only deal with *positional* strategies. Indeed, general strategies have to be represented by trees of arbitrary finite branching (so possibly different from D). Such trees can be representable in bounded FSO_D, but it seems that (without length comparison) we can not define in bounded FSO_D a formula relating a play in such a strategy to its underlying position in the input tree. This implies that correctness of such a general notion of acceptance can not be proved in bounded FSO_D.

On the other hand, *positional* strategies in acceptance games for D-ary trees can be described by D-ary trees labeled by functions from finite alphabets of "local moves". This allows to define a bounded FSO_D formula relating a play in a positional strategy to its underlying input tree position[6].

Another possibility would have been to use the more general Muller conditions, which are known to be determined with strategies of uniformly bounded finite memory. But formalizing this setting in bounded FSO_D (and proving determinacy) would have been significantly more complicated.

What we call a 'winning strategy' below corresponds, in fact, to this notion of positional winning strategy. We parametrize this notion by some winning condition $\mathcal{W}(V)$.

Definition IV.5 (Winning strategies)**.** *We define,*

$$\mathrm{WinStrat}(s, v, e, \mathcal{W})$$

as follows:

$$\begin{aligned} & s \text{ to } \mathsf{O}^{\mathsf{P}} \ \wedge \ \forall x \,.\, s(x) \ddot{\subseteq} \ e_{\mathsf{P}}(x) \\ \wedge \quad & \forall V \ddot{\ni} v \,.\, \mathrm{Play}(V, v, s \ddot{\cup} e_{\mathsf{O}}) \to \mathcal{W}(V) \end{aligned}$$

Notice that s is seen here as inducing a *subgame* of e, i.e. if $\mathrm{Game}(e, \mathsf{P}, \mathsf{O})$ then also $\mathrm{Game}(s \ddot{\cup} e_{\mathsf{O}}, \mathsf{P}, \mathsf{O})$, one in which there is always a unique choice of move by P.

Definition IV.6. *For an automaton $\mathcal{A}$ with acceptance condition Ω specified by an FSO_D formula $\mathcal{W}(V)$, we define,*

$$t \ddot{\in} \mathcal{L}(\mathcal{A}) : \quad \exists s \text{ to } \mathsf{O}^{\mathsf{P}} \,.\, \mathrm{WinStrat}(s, \iota, e^{\mathcal{A},t}, \mathcal{W})$$

where $\iota \doteq (\varepsilon, q^\iota)$.

D. Logical operations on automata in FSO

In this section, we present formalized constructions for the operations of complementation, finite union and projection. Finite union is, as usual, easy. Complementation relies on positional determinacy of (parity) acceptance games, whose formalized proof is deferred to Sect. VI.

[6]The complementation construction used in [15] relies on this.

For projection, we use the nondeterminization construction of [18]. It cleanly delimitates the powerset construction on alternating tree automata from the definition of the resulting acceptance condition, for which we rely on the embedding of MSO on ω-words in MSO_D provided by Prop. III.5.

We work in bounded FSO_D, relying on Th. III.13. We freely use the Recursion Theorem in bounded FSO_D (obtained, thanks to Thm. III.13, from the Recursion Theorem of MSO_D (Prop. III.3).

1) Singleton alphabet: We sketch a proof of the following:

Lemma IV.7. *If $\mathcal{A}$ is an automaton over the singleton alphabet, then $\mathsf{FSO}_D \vdash t \in \mathcal{L}(\mathcal{A})$ or $\mathsf{FSO}_D \vdash t \notin \mathcal{L}(\mathcal{A})$.*

The main point here is that it is easy to check whether automata without an input accepts or not. This can be seen as a simple case of the emptiness problem.

Any automaton over the singleton alphabet $\mathbf{2}^0 = \{\varepsilon\}$ can be rewritten to an automaton without an input, by replacing the input parameter for $\{\varepsilon\}^*$ and reducing as appropriate.

Now, from the point of view of the acceptance game, the opponent O does not any more care which direction he goes, since either way the node will be labeled by ε. From this point of view we can view the game in the following way,

$$\begin{array}{lcl} \mathsf{P}\text{-moves} & : & \{(q, \gamma) \mid \gamma \in \delta(q, \epsilon)\} \\ \mathsf{O}\text{-moves} & : & \{(\gamma, q) \mid (q, d) \in \gamma \text{ for some } d \in D.\} \end{array}$$

from any tree position.

Now, this game has only finitely many states and is positionally determined, by the usual argument. Moreover, there are only finitely many strategies and they can be checked for winningness by loop-detection in an exhaustion of its plays.

All of this can be represented by a finite quantifier-free formula over FSO_D and so the existence or otherwise of a winning strategy has a purely propositional proof in FSO_D.

2) Complement: Consider an automaton $\mathcal{A} = (Q, q^\iota, \delta, \mathcal{W})$[7] and define $\widetilde{\mathcal{A}} = (Q, q^\iota, \widetilde{\delta}, \widetilde{\mathcal{W}})$, where,

$$\begin{array}{lcl} \widetilde{\mathcal{W}}(V) & := & \neg \mathcal{W}(V) \\ \widetilde{\delta}(q, \mathsf{a}) & := & \{\gamma \in \mathcal{P}(Q \times D) \mid \gamma \cap \gamma' \neq \emptyset \text{ for any } \gamma' \in \delta\} \end{array}$$

We show in FSO_D that $\widetilde{\mathcal{A}}$ accepts the complement of the language computed by $\mathcal{A}$.

Lemma IV.8. $\mathsf{FSO}_D \vdash t \in \mathcal{L}(\widetilde{\mathcal{A}}) \longleftrightarrow t \notin \mathcal{L}(\mathcal{A})$.

Proof. This follows almost immediately from positional determinacy, Thm. VI.12. We need only to show that FSO_D proves that $e^{\widetilde{\mathcal{A}},t} \doteq \widetilde{e}^{\mathcal{A},t}$, which is a simple exercise. □

3) Union: Consider automata $\mathcal{A}_i = (Q_i, q_i^\iota, \delta_i, \mathcal{W}_i)$ for $i = 1, 2$. Define $\mathcal{A}_\cup := (Q_1 + Q_2 + \{q^\iota\}, q^\iota, \delta_\vee, \mathcal{W}_\vee)$, where, for $a \in \Sigma$, $\delta_\vee(q^\iota, a) := \delta_1(q_1^\iota, a) \cup \delta_1(q_2^\iota, a)$ and, for $q \in Q_i$, $\delta_\vee(q, a) := \delta_i(q, a)$, and moreover $\mathcal{W}_\vee(V) := \mathcal{W}_1(V) \vee \mathcal{W}_2(V)$.

[7]Throughout this section, we assume automata are equipped with an MSO_D winning condition, e.g. $\mathcal{W}(V)$ in this case.

Lemma IV.9. *The following is provable in* FSO_D.

$$t \in \mathcal{L}(\mathcal{A}_\cup) \longleftrightarrow (t \in \mathcal{L}(\mathcal{A}_1) \vee t \in \mathcal{L}(\mathcal{A}_2))$$

4) Projection: We now sketch the proof of the following:

Lemma IV.10. *For a parity automaton* $\mathcal{A}$*, there is a parity automaton* $\mathcal{A}^\exists$ *such that*

$$t \to \Sigma \times \{0,1\}, u \to \Sigma, (\pi_\Sigma t = u) \vdash u \in \mathcal{L}(\mathcal{A}^\exists) \longleftrightarrow t \in \mathcal{L}(\mathcal{A})$$

where $(\pi_\Sigma t = u)$ *stands for*

$$\forall x\, [(t(x) \doteq (u(x), 0)) \vee (t(x) \doteq (u(x), 1))]$$

We first translate a parity automaton to a nondeterministic automaton with an alternative acceptance condition, whence projection is simple to define, before converting back to a parity automaton.

a) Nondeterminization of an automaton: Consider and automaton $\mathcal{A} = (Q, q^\imath, \delta, \mathcal{W})$ on alphabet Σ. We first will build an equivalent *non-deterministic* automaton

$$\mathcal{A}^{\mathrm{ND}} \quad = \quad (\mathcal{P}(Q \times Q), \{(q^\imath, q^\imath)\}, \delta^{\mathrm{ND}}, \mathcal{W}^{\mathrm{ND}})$$

as follows. Consider $\boldsymbol{q} \in \mathcal{P}(Q \times Q)$ and $a \in \Sigma$ and write $\boldsymbol{q}\restriction 2 := \{q_1, \ldots, q_n\}$. Following [18], we let $\gamma^{\mathrm{ND}} \in \delta^{\mathrm{ND}}(\boldsymbol{q}, a)$ whenever there are $\gamma_1 \in \delta(q_1, a), \ldots, \gamma_n \in \delta(q_n, a)$ such that

$$\gamma^{\mathrm{ND}} \quad = \quad \{(\boldsymbol{q}_d, d) \mid d \in D\}$$

where, for each $d \in D$:

$$\boldsymbol{q}_d \quad = \quad \{(q_k, q) \mid 1 \leq k \leq n \text{ and } (q, d) \in \gamma_k\}$$

The acceptance condition $\mathcal{W}^{\mathrm{ND}}$ defines the set of sequences $(\boldsymbol{q}_n)_{n\in\mathbb{N}}$ such that for every $(q_n)_{n\in\mathbb{N}} \in Q^\omega$ with $\boldsymbol{q}_{n+1} = (q_n, q_{n+1})$, the sequence $(q_n)_{n\in\mathbb{N}}$ satisfies the parity condition of $\mathcal{A}$. This is definable by an FSO_D formula.

Note that $\mathcal{A}^{\mathrm{ND}}$ is non-deterministic, but in general not a parity automaton. By formalizing the argument of [18] (using the Recursion Theorem), we get:

Theorem IV.11. *For a parity automaton* $\mathcal{A}$*:*

$$\mathsf{FSO}_\mathbf{2} \vdash t \in \mathcal{L}(\mathcal{A}) \longleftrightarrow t \in \mathcal{L}(\mathcal{A}^{\mathrm{ND}})$$

Note that $\mathcal{A}^{\mathrm{ND}}$ does not a priori generates positionally determined games. However, since $\mathcal{A}$ is a parity automaton, a winning P strategy witnessing $t \in \mathcal{L}(\mathcal{A})$ can be assumed to be positional, and the the corresponding winning P strategy for $t \in \mathcal{L}(\mathcal{A}^{\mathrm{ND}})$ will be positional. In the context of Thm. IV.11, the converse direction follows from the fact that by definition of acceptance in our setting, $t \in \mathcal{L}(\mathcal{A}^{\mathrm{ND}})$ implies that there is a winning *positional* P strategy, which is then easily mapped to a winning positional P strategy witnessing $t \in \mathcal{L}(\mathcal{A})$.

b) Conversion to parity automaton: In order to maintain compatibility with the rest of our structural induction (e.g. closure under complements), we need to convert our nondeterministic automaton to one with a parity acceptance condition.

As noted in [18], the acceptance condition of $\mathcal{A}^{\mathrm{ND}}$ is an ω-regular language, which can thus be recognized by a deterministic parity ω-word automaton, say $\mathcal{D} = (Q_\mathcal{D}, q^\imath_\mathcal{D}, \delta_\mathcal{D}, \mathcal{W}_\mathcal{D})$ on the alphabet $\mathcal{P}(Q \times Q)$.

Following [18], we let

$$\mathcal{A}_{\mathrm{nd}} := (\mathcal{P}(Q \times Q) \times Q_\mathcal{D}, (q^\imath_{\mathcal{A}^{\mathrm{ND}}}, q^\imath_\mathcal{D}), \delta_{\mathrm{nd}}, \mathcal{W}_{\mathrm{nd}})$$

where $\delta_{\mathrm{nd}}((\boldsymbol{q}, q), a) := (\delta_{\mathcal{A}^{\mathrm{ND}}}(\boldsymbol{q}, a), \delta_\mathcal{D}(q, \delta_{\mathcal{A}^{\mathrm{ND}}}(\boldsymbol{q}, a)))$, and the parity condition $\mathcal{W}_{\mathrm{nd}}$ is defigned by assigning to state $(\boldsymbol{q}, q)$ the priority of q in $\mathcal{D}$.

Thanks to Prop. III.5, we can prove:

Proposition IV.12. *For a parity automaton* $\mathcal{A}$*:*

$$\mathsf{FSO}_\mathbf{2} \vdash t \in \mathcal{L}(\mathcal{A}) \longleftrightarrow t \in \mathcal{L}(\mathcal{A}_{\mathrm{nd}})$$

Lemma IV.10 is then obtained by the usual projection operation on non-deterministic automata, whose formalization is straightforward.

V. Completeness argument

This short Section gathers the results of the preceding Section to prove our main Thm. II.3. Recall that we proceed by formalizing a translation of formulas to automata. We begin by putting MSO_D formulas into a convenient form for this translation, namely with a purely relational vocabulary and only monadic variables. We then prove Thm. II.3.

A. Reduced syntaxes of MSO_D

For the translation of formulas to automata, it is and customary to work with formulas on a slightly different syntax.

1) Relational syntax: We first restrict to a purely relational vocabulary. It is based on the defined formulas $\mathrm{S}_d(x, y) := (\mathsf{S}_d(x) \doteq y)$ (for $d \in D$). The *relational formulas* $\phi, \psi \in \Lambda^R_D$ are build from atomic formulas Xy and $\mathrm{S}_d(x, yk)$ by means of $\neg$, $\vee$, $\exists x$ and $\exists X$.

To each formula $\phi \in \Lambda_D$ we associate a formula ϕ^R as follows. For a a term of MSO_D, define the formula $(z \triangleq t)$ by induction on t:

$$(z \triangleq y) := (z \doteq y) \qquad (z \triangleq \dot\varepsilon) := \neg\exists z' \bigvee_{d\in D} \mathrm{S}_d(z', z)$$
$$(z \triangleq \mathsf{S}_d(t)) := \exists z'[z' \triangleq a \wedge \mathrm{S}_d(z', z)]$$

Then, ϕ^R is obtained from ϕ by replacing each atomic Xa (with t not a variable) by $\exists z[(z \triangleq a) \wedge Xz]$, with z a fresh variable. Note that $\mathsf{MSO}_D \vdash (z \triangleq a) \longleftrightarrow (z \doteq a)$.

Lemma V.1. *For every formula* $\phi \in \Lambda_D$*, we have* $\mathsf{MSO}_D \vdash (\phi \longleftrightarrow \phi^R)$.

2) Individual-free syntax: The next step is to get rid of individual quantifiers. Consider the defined formulas

$$\begin{array}{lcl}(X \mathrel{\dot\subseteq} Y) & := & \forall x(Xx \to Yx)\\ \mathrm{S}_d(X,Y) & := & \exists xy\,[Xx \wedge Yy \wedge \mathrm{S}_d(x,y)]\end{array}$$

The *individual-free formulas* $\phi, \psi \in \Lambda_D^0$ are build from atomic formulas $(X \mathrel{\dot\subseteq} Y)$ and $\mathrm{S}_d(X,Y)$ by means of negation, disjunction and monadic second-order quantification $\exists X$ only.

Let $\phi \in \Lambda_D^R$ with $\mathrm{FV}(\phi) = \{x_1, \ldots, x_p, Y_1, \ldots, Y_q\}$. We inductively associate to ϕ a formula $\phi^0 \in \Lambda_D^0$ with free variables $\{X_1, \ldots, X_p, Y_1, \ldots, Y_q\}$ as follows: we let $(\exists x_{p+1}\phi)^0 := \exists X_{p+1}\left[\mathrm{Sing}(X_{p+1}) \wedge \phi^0\right]$ and

$$\begin{array}{rclcrcl}(Y_j x_i)^0 & := & X_i \mathrel{\dot\subseteq} Y_j & \qquad & \mathrm{S}_d(x_i,x_j) & := & \mathrm{S}_d(X_i,X_j)\\ (\neg\phi)^0 & := & \neg\phi^0 & & (\phi\vee\psi)^0 & := & \phi^0 \vee \psi^0\\ (\exists Y_{q+1}\phi)^0 & := & \exists Y_{q+1}\,\phi^0 & & & & \end{array}$$

where, with $(X \doteq \emptyset) := \forall Y(X \mathrel{\dot\subseteq} Y)$, $\mathrm{Sing}(X)$ is defined as

$$\neg(X \doteq \emptyset) \wedge \forall Y\left[Y \mathrel{\dot\subseteq} X \to \left(Y \doteq \emptyset \vee X \mathrel{\dot\subseteq} Y\right)\right]$$

Lemma V.2. *For every formula $\phi \in \Lambda_D^R$ with $\mathrm{FV}(\phi) = \{\vec{x}, \vec{Y}\}$, we have $\vec{Xx}, \mathrm{Sing}(\vec{X}) \vdash_{\mathsf{MSO}_D} \left(\phi \longleftrightarrow \phi^0\right)$.*

Putting everything together we have:

Corollary V.3. *For every closed formula $\phi \in \Lambda_D$, there is a closed formula $\psi \in \Lambda_D^0$ such that $\mathsf{MSO}_D \vdash (\phi \longleftrightarrow \psi)$.*

B. From formulas to automata

We now give the interpretation of formulas to automata. To each formula $\phi \in \Lambda_D^0$ and sequence of distinct monadic variables $X_1, \ldots, X_n$ such that $\mathrm{FV}(\phi) \subseteq \{X_1, \ldots, X_n\}$ we associate an automata by induction on ϕ as usual.

For the atomic formulas of Λ_D^0, we use the following facts:

Lemma V.4. *Given $X_1, \ldots, X_n$ and $i, j \in \{1, \ldots, n\}$, there is an automaton $\mathcal{A}$ on the alphabet $\mathbf{2}^n$ such that*

$$\mathsf{FSO}_D \vdash \forall X_1, \ldots, X_n \left((X_i \mathrel{\dot\subseteq} X_j) \longleftrightarrow \mathcal{L}(\mathcal{A})[X_1, \ldots, X_n]\right)$$

Lemma V.5. *Given $X_1, \ldots, X_n$, $i, j \in \{1, \ldots, n\}$ and $d_0 \in D$, there is an automaton $\mathcal{B}$ on the alphabet $\mathbf{2}^n$ such that*

$$\mathsf{MSO}_D \vdash \forall X_1, \ldots, X_n \left(\mathrm{S}_{d_0}(X_i, X_j) \longleftrightarrow \mathcal{L}(\mathcal{B})[X_1, \ldots, X_n]\right)$$

Proposition V.6. *Consider a formula $\phi \in \Lambda_D^0$ and a sequence of distinct monadic variables $X_1, \ldots, X_n$ containing the free variables of ϕ.*

There is an automaton $\mathcal{A}$ on the alphabet $\mathbf{2}^n$ such that

$$\mathsf{MSO}_D \vdash \forall X_1, \ldots, X_n \left(\phi \longleftrightarrow \mathcal{L}(\mathcal{A})[X_1, \ldots, X_n]\right)$$

Proof. For the atomic formulas we use Lemmas V.4 and V.5 respectively. For the logical connectives $\neg$, $\vee$ and $\exists X$, we use Lemmas IV.8, IV.9 and IV.10 respectively. At each step we rely on Thm. III.13 for the interpretation of bounded FSO_D formulas in MSO_D. □

C. Completeness of MSO_D

We can now prove Theorem. II.3.

Proof of Thm. II.3. Consider a closed formula $\phi \in \Lambda_D$. By Cor. V.3, there is a closed formula of $\psi \in \Lambda_D^0$ such that

$$\mathsf{MSO}_D \vdash \phi \longleftrightarrow \psi$$

Then, thanks to Prop. V.6 there is an automaton $\mathcal{A}$ on the singleton alphabet such that

$$\mathsf{MSO}_D \vdash \phi \longleftrightarrow \mathcal{L}(\mathcal{A})$$

But now, by Lem. IV.7 we have either $\mathsf{MSO}_D \vdash \mathcal{L}(\mathcal{A})$ or $\mathsf{MSO}_D \vdash \neg\mathcal{L}(\mathcal{A})$. (Note that since $\mathcal{A}$ is an automaton on the alphabet $\mathbf{2}^0$, the FSO_D formulas $t \in \mathcal{L}(\mathcal{A})$ and $t \notin \mathcal{L}(\mathcal{A})$ expand to the MSO_D formulas $\mathcal{L}(\mathcal{A})$ and $\neg\mathcal{L}(\mathcal{A})$ respectively.) Hence either $\mathsf{MSO}_D \vdash \phi$ or $\mathsf{MSO}_D \vdash \neg\phi$ □

VI. Positional determinacy

The goal of this section is to prove that abstract games, as defined in our setting, with parity winning conditions, are positionally determined, provably in FSO_D and so also MSO_D. This result is critical to show that the languages recognized by alternating tree automata are closed under complement, cf. Lemma IV.8.

Throughout this section we will work inside an arbitrary Henkin model of FSO_D unless otherwise stated.

A. Topology of abstract games

Before we proceed we need to introduce some terminology on the *topology* of abstract games. We express all concepts from the point of view of player, P, for simplicity. This is adequate since we will be able to switch to a *dual* game where the roles of P and O are morally switched.

In line with the notational conventions of Def. IV.1 and IV.2, we use subscripts P and O on variables for sets of game positions to denote the subsets of P-positions and O-positions respectively. I.e., for a variable V, we apply Comprehension to define:[8]

$$\begin{array}{rcl}(x,a) \mathrel{\dot\in} V_{\mathsf{P}} & \longleftrightarrow & a \mathrel{\ddot\in} \mathsf{P} \wedge (x,a) \mathrel{\dot\in} V\\ (x,a) \mathrel{\dot\in} V_{\mathsf{O}} & \longleftrightarrow & a \mathrel{\ddot\in} \mathsf{O} \wedge (x,a) \mathrel{\dot\in} V\end{array}$$

Of course, as one would expect, we have that $V \doteq V_{\mathsf{P}} \mathbin{\dot\cup} V_{\mathsf{O}}$.

Definition VI.1 (Subsets of game positions)**.** *We define the* complement $\widetilde{V}$ *of a set of game positions V by Comprehension as follows:*

$$(x,a) \mathrel{\dot\in} \widetilde{V} \longleftrightarrow (a \mathrel{\ddot\in} \mathsf{P} \sqcup \mathsf{O} \wedge \neg(x,a) \mathrel{\dot\in} V)$$

We define the reachability *condition with respect to U:*

$$\mathrm{Reach}^U(V) := \quad \exists v.v \mathrel{\dot\in} U \wedge v \mathrel{\dot\in} V$$

For a winning condition $\mathcal{W}(V)$ and game e we define the winning set $V^{e,\mathcal{W}}$ *by Comprehension as follows:*

$$v \mathrel{\dot\in} V^{e,\mathcal{W}} \longleftrightarrow \exists s \text{ to } \mathsf{O}^{\mathsf{P}} \,.\, \mathrm{WinStrat}(s, v, e, \mathcal{W})$$

[8]Here we are technically applying Comprehension for function and set variables, due to the translation of variables for game positions.

Finally, a trap V *under a game* e *is defined as follows:*

$$\mathrm{Trap}(V,e): \begin{array}{ll} & \forall u \mathrel{\dot\in} V_{\mathsf{P}} \,.\, \forall v \,.\, u \xrightarrow[e_{\mathsf{P}}]{} v \;\to\; v \mathrel{\dot\in} V_{\mathsf{O}} \\ \wedge & \forall u \in V_{\mathsf{O}} \,.\, \exists v \,.\, u \xrightarrow[e_{\mathsf{O}}]{} v \;\wedge\; v \mathrel{\dot\in} V_{\mathsf{P}} \end{array}$$

An important observation is that, in deadlock-free games, traps induce deadlock-free subgames: each player always has a move to make staying in the trap.

Proposition VI.2. FSO_D *proves the following:*

$$\begin{array}{ll} & \exists U.\mathrm{Play}(U,v,e) \wedge \mathrm{Trap}(V,e) \wedge v \in V \\ \to & \exists U' \subseteq V.\mathrm{Play}(U',v,e) \end{array}$$

Let us denote the corresponding subgame by $e{\restriction}V$, obtained by comprehension as follows:

$$u \xrightarrow[e{\restriction}V]{} v \;\longleftrightarrow\; u \mathrel{\dot\in} V \wedge v \mathrel{\dot\in} V \wedge u \xrightarrow[e]{} v \tag{8}$$

We also have that complements of winning sets under reachability are traps.

Proposition VI.3. $\mathsf{FSO}_D \vdash \mathrm{Trap}(\widetilde{V}^{e,\mathrm{Reach}^U}, e)$

We now wish to show that plays differing on only finitely many game positions are equi-winning. As a consequence we have that parity winning sets are closed under reachability.

Definition VI.4. *Write* $X \mathrel{\dot\sim} Y$ *if* X *and* Y *differ on only a finite set, i.e.:*

$$\exists x.\forall y \mathrel{\dot\geq} x.y \mathrel{\dot\in} X \longleftrightarrow y \mathrel{\dot\in} Y$$

We extend this definition to sets of game positions, writing $(X,f) \mathrel{\dot\sim} (Y,g)$ *for:*

$$X \mathrel{\dot\sim} Y \;\wedge\; \exists x \mathrel{\dot\in} X \mathrel{\dot\cap} Y \forall y \mathrel{\dot\geq} x.f(y) \mathrel{\ddot=} g(y)$$

Lemma VI.5. $\mathsf{FSO}_D \vdash (U \sim V \wedge \mathrm{Par}^c(U)) \to \mathrm{Par}^c(V)$.

We can use this lemma to show that winning sets under parity conditions are already winning sets under reachibility conditions, helping us to induce subgames later.

Theorem VI.6. $\mathsf{FSO}_D \vdash V^{e,\mathrm{Par}^c} \doteq V^{e,\mathrm{Reach}^{V^{e,\mathrm{Par}^c}}}$.

B. Main result

We give the main argument of positional determinacy, assuming the existence of *uniform* winning strategies, as stated in the following result.

Theorem VI.7 (Uniformity)**.** FSO_D *proves the following,*

$$\exists s \text{ to } \mathsf{O}^{\mathsf{P}}.\forall v \in V^{e,\mathcal{W}}.\mathrm{WinStrat}(s,v,e,\mathcal{W})$$

for $\mathcal{W} \in \{\mathrm{Par}, \mathrm{Reach}\}$.

We give a construction of such strategies and a proof that they are winning on the winning set in the following section.

Definition VI.8 (Dual and union games)**.** *Let* e, e' *be games on labels* P *and* O*, we define the* dual *game* $\widetilde{e}$ *by Comprehension as follows:*

$$\begin{array}{rcl} \widetilde{e}_{\mathsf{P}}(x) & \ddot= & \{(q,\gamma) \in \mathsf{P} \times \mathsf{O} : (q,\gamma') \mathrel{\ddot\in} e_{\mathsf{P}}(x) \Rightarrow \gamma \cap \gamma' \neq \emptyset\} \\ \widetilde{e}_{\mathsf{O}}(x) & \ddot= & \{(d,\gamma,q) \in D \times \mathsf{O} \times \mathsf{P} : (q,d) \in \gamma\} \\ \widetilde{e}(x) & \ddot= & \widetilde{e}_{\mathsf{P}}(x) \mathrel{\ddot\cup} \widetilde{e}_{\mathsf{O}}(x) \end{array}$$

We also define the union *game* $e \mathrel{\ddot\cup} e'$ *by Comprehension:*

$$(e \mathrel{\ddot\cup} e')(x) = e(x) \mathrel{\ddot\cup} e'(x)$$

Our notion of duality commutes with arbitrary restrictions:

Proposition VI.9. $\mathsf{FSO}_D \vdash \widetilde{e{\restriction}V} \mathrel{\dot=} \widetilde{e}{\restriction}V$.

In the general case, for arbitrary V, the restricted games might not be deadlock-free.

Definition VI.10 (Dual and equivalent parity conditions)**.** *Let* $c, c' : \mathsf{P} \to [0,n]$ *be parity functions. We define* $\widetilde{c}(q) = c(q)+1$.

We say that c *and* c' *are* equivalent *if:*

1) $c(q) < c(q')$ *iff* $c'(q) < c'(q')$.
2) $c(q)$ *is even iff* $c'(q)$ *is even.*

Proposition VI.11. *Let* c *be a parity function* $\mathsf{P} \to [0,n]$*. We have that* $\mathsf{FSO}_D \vdash \mathrm{Par}^c(V) \longleftrightarrow \neg\mathrm{Par}^{\widetilde{c}}(V)$.

If c' *is a parity function equivalent to* c *then* $\mathsf{FSO}_D \vdash \mathrm{Par}^c(V) \longleftrightarrow \mathrm{Par}^{c'}(V)$.

In particular note that, by the above proposition, we can assume without loss of generality that each parity function has range $[0,n)$ or $(0,n]$ for some n.

We now give a proof of our positional determinacy theorem. Due to the invariants required, we show positional determinacy of a parity game starting from any position.

Theorem VI.12. FSO_D *proves the following:*

$$\forall v. \left(\begin{array}{ll} & \exists s.\mathrm{WinStrat}(v,s,e,\mathrm{Par}^c) \\ \vee & \exists s.\mathrm{WinStrat}(v,s,\widetilde{e},\mathrm{Par}^{\widetilde{c}}) \end{array} \right)$$

Our argument essentially formalizes that of [15], by an external induction on the size of the range of c, differing only in the later construction of uniform strategies.

C. Construction of uniform strategies

In this section we give a construction of the uniform winning strategies that we required for the positional determinacy proof, Thm. VI.12. In general, for parity games, this seems to require the axiom of choice or transfinite well-orderings, generally unavailable in FSO_D and MSO_D, however we exploit the fact that there is already plenty of order structure in our underlying game graph.

For the following we fix a well-order, $\leq$, on O^{P} and parametrise our results by some winning condition $\mathcal{W} \in \{\mathrm{Par}, \mathrm{Reach}\}$ on a game e.

Definition VI.13 (Ordering strategies)**.** *For an individual symbol* x*, we define,*

$$s \mathrel{\dot\leq_x} s'$$

as:

$$\exists y \mathrel{\dot\leq} x.\exists z \mathrel{\ddot\in} \mathsf{P} \sqcup \mathsf{O}.\forall y' \mathrel{\dot\leq} x.\forall z' \mathrel{\ddot\in} \mathsf{P} \sqcup \mathsf{O}. \left[\begin{array}{ll} & \left(\begin{array}{ll} & \mathrm{WinStrat}(s,(y,z),e,\mathcal{W}) \\ \wedge & \mathrm{WinStrat}(s',(y',z'),e,\mathcal{W}) \end{array} \right) \\ \to & y \mathrel{\dot<} y' \vee (y \mathrel{\dot=} y' \wedge s(y) \mathrel{\ddot\leq} s'(y)) \end{array} \right]$$

We also define a 'least' strategy s *from a position* x*,*

$$\mathrm{Least}_x(s,\phi,f)$$

as:

$$\phi[f/s] \wedge (\forall g \ . \ \phi[g/s] \to f \ \dot{\leq}_x \ g)$$

Finally we give a definition of the uniform strategy from a winning position.

Definition VI.14 (Uniform strategies)**.** *We define* $s^{\mathrm{un}}(x) = y$ *by Comprehension as follows:*

$$\begin{array}{ll} & y \ddot{\in} \mathsf{O}^{\mathsf{P}} \\ \wedge & \forall z \ddot{\in} \mathsf{P} \ . \ \forall f \ . \\ & \left(\begin{array}{ll} & \mathrm{Least}_x(s, \mathrm{WinStrat}(s,(x,z),e,\mathcal{W}), f) \\ \to & y(z) \ddot{=} f(z) \end{array} \right) \end{array}$$

This construction of uniform strategies rests on the fact that acceptance games are well-founded. The idea here is that a "minimal" winning strategy at some game position x is a strategy such that the first position from which this strategy is winning is minimal among positions from which winning strategies at position x are winning. When two such minimal strategies are in conflict, we follow a finitely generated order on games positions. Now, by always following a minimal strategy in this sense, the position witnessing that this strategy is minimal can only decrease in the game. Hence we eventually converge to a play of a winning strategy.

Theorem VI.15. FSO_D *proves the following:*

$$v \ddot{\in} V^{e,\mathcal{W}} \to \mathrm{WinStrat}(s^{\mathrm{un}}, v, e, \mathcal{W}) \tag{9}$$

VII. Further work

A. On the proof-theoretic strength of MSO

An important future work is the calibration of the proof theoretic strength of MSO as a subsystem of PA2. It is customary to compare subsystems of PA2 obtained by restricting the Comprehension scheme to formulas of given logical complexity [13][9].

We conjecture that the axiomatization of MSO on ω-words can be carried over in a system called *Weak Koening Lemma*. For the case of trees, the topological complexity of MSO suggests that at least Δ^1_2-Comprehension is required.

We point out that perhaps our most interesting use of Comprehension was in the definition of $\dot{\leq}_x$ in Def. VI.13, was on a Σ^1_2-MSO formula (under the interpretation). Interestingly, this formula is in fact Δ^1_2 in MSO, and so we suspect that Δ^1_2-Comprehension in MSO_D suffices to carry out our proof.

B. Some remarks on proof search

One outcome of this work is the possibility to implement decision procedures for SnS based on proof search. Our main result, the complete axiomatization, can be seen as a 'proof of concept' for this approach:

Algorithm VII.1. *Under an input closed* MSO_D*-formula* ϕ*, enumerate all* MSO_D*-proofs until one with conclusion* ϕ *or* $\neg\phi$ *is reached.*

[9]For this to make sense, one works with primitive equality.

Of course, this is not a very sophisticated algorithm, and it is worth restating that the its correctness is itself due to the usual automata-logics argument. However the algorithm, nonetheless, makes no mention of automata and so can be adapted and improved purely in the setting of proof theory. In this sense, this algorithm is the first of its kind: a decision procedure for SnS that remains internal to the language.

We state some ideas in this section for future work on how to adapt this proof-search for MSO. They are fundamentally linked to notions of proof-theoretic strength explained above.

a) Complexity of nonlogical rules: At the heart of the difficulty of proof search is the complexity of Induction and Comprehension formulae used, i.e. proof-theoretic strength as we previously discussed.[10] This is due to the free-cut elimination theorem, which states that a proof can be transformed into one where all formulae occurring are subformulae[11] of the conclusion, an axiom, or a nonlogical step.

b) Towards a terminating bottom-up proof search procedure: An interesting course of future work would be to conduct a bona fide proof search procedure, starting from the conclusion and building the proof bottom-up.

Progress towards this might be made by the following related goals:

1) Eliminate as many cuts as possible.
2) Eliminate as many nonlogical steps as possible.

One avenue worth considering is that there are implementations of comprehension which admits cut-reduction steps. Usually the problem with this approach is that these steps do not terminate, but due to the inherent decidability of SnS, and termination of proof-search in MSO_D, perhaps there is some hope to adapt the cut-reductions to achieve termination. Of course, even if this were possible, there would still be the problem of induction steps but, as previously mentioned, these do not contribute much to the quantifier-complexity of proofs in our setting.

One could also choose to work in the FSO setting. Here, since Choice is valid in the standard model (and indeed an axiom of FSO), one can apply choice *eagerly* via Skolemization, eliminating the consideration of where it might occur in a proof. The trade off here is that we might increase quantifier complexity, but this is a trade off worth exploring.

c) Proof interaction and verification: Even without any improvement on the proof search algorithm, the very presence of a proof theory admits the possibility of *interactive* proofs, ones where a user may state as much or as little of the proof information to a proof assistant. For example, if the user gives all comprehension formulae then the rest of the proof can be reconstructed bottom-up quite efficiently, as mentioned in the previous section.

In the same vein, a complete axiomatization provides a mean to *communicate* proofs that can be checked quickly, rather than having to redecide formulae. Such a tool might be

[10]Due to our formulation of induction, however, this complexity is devolved to just the comprehension instances.

[11]In the usual wide sense.

useful for *cyclic* theorem provers, where correctness criteria boil down to checking the inclusion between Büchi automata, what could be an incredibly complex procedure.

VIII. Concluding remarks

We presented a completed axiomatization of MSO on D-ary ary trees, for D an arbitrary non-empty finite set. Our axiomatization is a subsystem of PA2. This provide an interesting basis to study proof-theoretical aspects of MSO, and also suggests new algorithms to decide MSO formulas, which will be the object of future investigations.

Acknowledgments

Early stages of this work greatly benefited from the participation of Alexander Kreuzer, who suggested a formalization of the algebraic approach of [1] instead of the translation of formulas to automata. Such algebraic tools might be useful for further work on proof-theoretic complexity and for the axiomatization of MSO on the countably branching tree $\mathbb{N}^*$.

This work also benefited from regular discussions with Thomas Colcombet and Arnaud Carayol.

References

[1] A. Blumensath, "An algebraic proof of Rabin's Tree Theorem," *Theor. Comput. Sci.*, vol. 478, pp. 1–21, 2013.

[2] J. R. Büchi and D. Siefkes, "Axiomatization of the Monadic Second Order Theory of ω_1," in *Decidable Theories II : The Monadic Second Order Theory of All Countable Ordinals*, ser. LNM, J. R. Büchi and D. Siefkes, Eds. Springer, 1973, vol. 328, pp. 129–217.

[3] A. Carayol and C. Löding, "MSO on the Infinite Binary Tree: Choice and Order," in *CSL*, ser. Lecture Notes in Computer Science, vol. 4646. Springer, 2007, pp. 161–176.

[4] A. Das and C. Riba, "A complete axiomatization of mso on infinite trees," 2015, available at http://www.anupamdas.com/msotree-full.pdf.

[5] E. A. Emerson and C. S. Jutla, "Tree Automata, Mu-Calculus and Determinacy (Extended Abstract)," in *FOCS*. IEEE Computer Society, 1991, pp. 368–377.

[6] A. Gheerbrant and B. ten Cate, "Complete Axiomatizations of Fragments of Monadic Second-Order Logic on Finite Trees," *Logical Methods in Computer Science*, vol. 8, no. 4, 2012.

[7] E. Grädel, W. Thomas, and T. Wilke, Eds., *Automata, Logics, and Infinite Games: A Guide to Current Research*, ser. Lecture Notes in Computer Science, vol. 2500. Springer, 2002.

[8] S. Gurevich and S. Shelah, "Rabin's Uniformization Problem," *J. Symb. Log.*, vol. 48, no. 4, pp. 1105–1119, 1983.

[9] D. E. Muller and P. E. Schupp, "Simulating Alternating Tree Automata by Nondeterministic Automata: New Results and New Proofs of the Theorems of Rabin, McNaughton and Safra," *Theor. Comput. Sci.*, vol. 141, no. 1&2, pp. 69–107, 1995.

[10] M. O. Rabin, "Decidability of Second-Order Theories and Automata on Infinite Trees," *Transactions of the American Mathematical Society*, vol. 141, pp. 1–35, 1969.

[11] C. Riba, "A model theoretic proof of completeness of an axiomatization of monadic second-order logic on infinite words," in *Proceedings of IFIP-TCS'12*, 2012.

[12] D. Siefkes, *Decidable Theories I : Büchi's Monadic Second Order Successor Arithmetic*, ser. LNM. Springer, 1970, vol. 120.

[13] S. Simpson, *Subsystems of Second Order Arithmetic*, 2nd ed., ser. Perspectives in Logic. Cambridge University Press, 2010.

[14] B. ten Cate and G. Fontaine, "An Easy Completeness Proof for the Modal μ-Calculus on Finite Trees," in *FOSSACS*, ser. Lecture Notes in Computer Science, vol. 6014. Springer, 2010, pp. 161–175.

[15] W. Thomas, "Languages, Automata, and Logic," in *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa, Eds. Springer, 1997, vol. III, pp. 389–455.

[16] D. van Dalen, *Logic and Structure*, 4th ed., ser. Universitext. Springer, 2004.

[17] I. Walukiewicz, "Completeness of Kozen's Axiomatisation of the Propositional μ-Calculus," *Information and Computation*, vol. 157, no. 1-2, pp. 142–182, 2000.

[18] ——, "Monadic second-order logic on tree-like structures," *Theor. Comput. Sci.*, vol. 275, no. 1-2, pp. 311–346, 2002.