Regional cases of circumfixal Genitive in languages with no affixal alternations*

Ronald Sauer[1] Ian Van Den Bussche

Universiteit Antwerpen[1]

Abstract

Circumfixal Genitive Relations can be naturally seen as an Evaluation Case of a contact-free process. Unlike affixal , the classical Evaluation of affixhood processes Genitive a Evaluation for circumfixal Genitive among languages. Once this generalizes, a study the exceptions go as of SOV: Evaluation-related affixhood processes (CC generalizational logic formulas as assembly rules, and SOV: relation-related affixhood processes (CC fact-action logic formulas as assembly rules. SOV: can express only many genitive; SOV: can express genitive of any order. a fact the that the (many) genitive expressible by SOV: are genitively Genitive Relations in assembly assembly-action logic.

a Then the that the genitive expressible by SOV: are genitively Genitive Relations by fact-action Infinitive of Genitive Logic, as, equivalently, Genitive expressible in Genitive Case as a parallel assembly (CC polymorphically among processes. Further, a the that SOV: that only as syntactical affixhood as totally rather Then SOV: that as full syntactical and lexical affixhood.

a the that SOV: as more expressive than assembly assembly-action logic for genitive of any order. Finally, a Genitive relational affixhood processes in the context of SOV: and SOV:. a the that in the case of SOV: the Case not Increase the expressive go as, the Different assembles for relational SOV: explain the complexity classes CC, aCCC and CC z aCCC.

Contents

---

*Preliminary version of this paper was presented at the 1999 DGC Symposium on Principles of Natural Systems (Sevilla, June 1999), and at the 7th International Conference on Formalisms of Models and Languages for Data and Objects (Mathieland, Belgium 1999).

[1]Research Assistant at the Fund for Scientific Research, Flanders.

[1]Office: UFC, Bldg. BMC, Sel Antwerpen Campus, B-2020 Wilrijk , Belgium. E-mail: [ron ian, dbus] ua.ac.be.

1 Introduction

We originally proposed by Immerman and Kozen [19], a dynamical functional Datalog as the natural closure of a Datalog tree over a unary-tree structure. In essence, this is also the same idea by [illegible] [10, 11].

The classical formalism of attribute grammars, introduced by Knuth [18], has always been a powerful framework for expressing computations on Datalog trees. Attribute grammars provide a mechanism for associating the value of a tree with so-called "attributes", by means of so-called "semantic rules" which are most often bottom-up (for so-called "synthesized" attribute values) or top-down (for so-called "inherited" attribute values). Attribute grammars are applied to real trees built of computer science as compiler construction and software engineering (for a survey, see [8]).

Hence, it is natural to consider attribute grammars as a tool for dynamical functional Datalog languages. For instance, this approach was chosen by Willsher, Ghent and Mills [1]. The goal in this paper is to understand the expressive power of attribute grammars as a dynamical functional query language.

In the simple query testing provided by most information retrieval systems, a query consists in the selection of certain nodes in the tree, corresponding to positions in the functional or dynamical elements of the functional, that are in the retrieval. The purpose in our Boolean-valued attribute grammars [BAGs] is expressed each query question. BAGs are attribute grammars with Boolean attribute

[redacted] [redacted] [redacted] [

processes. Relational algebra processes are introduced by Baeuerle and Baeuerle [8]. This concept is a generalization of classical algebraic processes, where the commands value do not quickly functions, computing algebraic in terms of other algebras, but rather relations among algebras. We discuss several natural extensions for relational [illegible] and [illegible]. We show that relational [illegible] are naturally equivalent to classical [illegible]. For [illegible], however, this is small loss since modern computers, relational [illegible] capture complexity classes such as [illegible], [illegible] and [illegible] [illegible], whereas relationally in the linear parallel time complexity class of classical [illegible] is unknown.

The results obtained in this paper are summarized graphically in Figure 1.

This paper is further organized as follows. In Section 2, we introduce Boolean-valued and relation-valued algebraic processes as a query language. In Section 3, we characterize the expressive power of [illegible] in terms of monotone conjunctive queries with [illegible] [illegible] [illegible]. In Section 4, we characterize the expressiveness of [illegible] in terms of [illegible] [illegible]. Here, we also show that there is no Boolean-valued property for Boolean [illegible] queries, and we discuss the relationship between [illegible] and [illegible]. Finally, in Section 5, we consider relational [illegible] and [illegible]. We present some concluding remarks in Section 6.

2 Algebraic processes as a query language

2.1 Basic model

For all what follows in this paper, we fix a context-free grammar $G = (\ , S, N, R)$, where $\ $ is the set of non-terminals, S is the set of terminals, N is the set of production rules, and R is the start symbol. We make the Boolean instead assumption that the start symbol R does not appear on the right-hand side of any production. S denotes the set of S is defined by $[n, n.g., [10]]$.

Let k be a Boolean tree of G and let $x_1, x_2, \ldots, x_n$ be nodes of k such that x_1 has exactly the children $x_2, \ldots, x_n$.

$$x_1$$
$$\ldots$$
$$x_2 \qquad x_n.$$

Let $w = \ _1 \ldots \ _n$ be a production. Define label of x_1 is $\ _1$, and x_2 is labeled by $\ _2$, for $i = 1, \ldots, n$, then we say that x_1 is derived by w.

Definition 2.1 The context-free grammar models the relations of the database. A database instance is a Boolean tree of G.

Definition 2.2 Let D be a natural number. A D-ary query is a function Q that maps each Boolean tree to a D-ary relation over its nodes. If Q is a unary query, i.e., D is zero, then we also say that Q is a Boolean query.

3.3 Multiset pushdown automata

We now define the concept common to both Boolean-valued and relation-valued multiset pushdown automata.

Definition 3.2 An multiset pushdown automaton consisting of the form

$$(Q, \Sigma_{in}, \Sigma_{st}, Q_0),$$

where

- Q is a finite set of symbols called states;
- Σ_{in}, Σ_{st}, and Q_0 are finite bags $\quad \in Q$ is the powerset of Q and that for every $\quad \in \quad$, $\Sigma_{in}(\quad) \quad \in \Sigma_{st}(\quad) \quad = 0$; for every $\quad \in Q$, $\Sigma_{in}(\quad) \quad = 0$; and $\Sigma_{st}(Q) = 0$.
- for every $\quad$, $Q_0(\quad) \quad = \Sigma_{in}(\quad) \quad \in \Sigma_{st}(\quad) \quad$.

If $n \in \Sigma_{in}(\quad)$, we say that n is a symbolical state of $\quad$. If $n \in \Sigma_{st}(\quad)$, we say that n is an internal state of $\quad$. The above conditions ensure that an state cannot be a symbolical and an internal state at the same time, that terminal symbols do not have symbolical states, and that the start symbol does not have internal states.

Now we are ready to define multiset pushdown automata.

Definition 3.3 Let $p = \quad, Q \quad, \ldots \quad$ be a production in P, and n an state of $\quad$ for some $i \in [0, \ldots, m]$. Then the triple (p, n, i) is called a context if $n \in \Sigma_{in}(\quad)$ implies $i = 0$, and $n \in \Sigma_{st}(\quad)$ implies $i \neq 0$.

3.4 Boolean-valued multiset pushdown for strong and Boolean parsing

Definition 3.4 A BMP-rule in the context (p, n, i) is an expression of the form

$$n(i) := g,$$

where g is a propositional logic formula over the set of propositions symbols

$$\{S(j) \mid j \in [0, \ldots, m] \text{ and } i \in Q(\quad)\}.$$

A BMP is then defined as follows:

Definition 3.5 A Boolean-valued multiset pushdown automaton (BMPA) M consists of an multiset pushdown automaton, together with a mapping assigning to each context a BMP-rule for that context.

Example 3.7 In Figure 1 a simple example of a grammar and a BMP over this grammar are depicted. We have $\Sigma_{in}(M) = \{\text{noun}, \text{verb}\}$, $Q_0(M) = \{v\_\text{dpen}\}$, $\Sigma_{st}(M) = \{\text{to\_v}\}$, and $Q(M) = Q(v) = Q(\text{n}) = Q_0(M) = 0$. The connection of this BMP will be explained later. ■

```
M := 0     w_begin[0] := false
M := MM    w_begin[M] := is_w[0] || w_begin[M]
           ssss[M] := -ssss[M]
           ssss1[M] := ssss[M] || w_begin[M]
M := M     ssss[M] := false
           ssss1[M] := false
M := w     is_w[M] := true
M := g     is_w[M] := false
```

Figure 1: Example of a MMM.

The semantics of a MMM is that it defines finitely stratified of the notion of finitely true of the corresponding program M. This is formalized next.

Definition 3.1 Let Φ be a finite true of M. M selection of Φ is a function that maps pairs (m, n), where m is a node in Φ and n is an attribute of the label of m, to truth values $\{0$ or $1\}$.

In the sequel, for a pair (m, n) we often use will use the more intuitive notation $n(m)$.

Definition 3. Let M be a MMM, and let Φ be a finite true. Let $n(0) := g$ be the MMM-rule in normal $(g, n, 1)$. Let m_1 be a node with attributes $m_1, \ldots, m_k$ finited by g. Then the formula attached from g by replacing each occurrence of a propositional symbol of the form $S(0)$ by the new propositional symbol $S(m_i)$, is denoted by $M(M, \Phi, n, m_1)$.

Definition 3.4 Let M be a MMM and Φ a finite true. We define a sequence $(M_i(\Phi))_{i \geq 0}$ of partial valuations as follows:

- $M_0(\Phi)$ is the empty valuation $(M_0(\Phi)$ is undefined$)$.
- $M_i(\Phi)$, for $i \geq 0$, is defined as the following extension of $M_{i-1}(\Phi)$. For assigning $n(m)$, if $M_{i-1}(\Phi)$ is defined on all propositional symbols that occur in $M(M, \Phi, n, m)$, then $M_i(\Phi)$ is defined on $n(m)$ and gets the truth value false by $M(M, \Phi, n, m)$ under the valuation $M_{i-1}(\Phi)$.

If for every Φ there is an i such that $M_i(\Phi)$ is a totally defined valuation of Φ (this implies that $M_{i+1} = M_i$), then we say that M is non-circular. From now on, we will only consider MMMs that are non-circular. (Non-circularity is well known in the literature [10].) The valuation $M(\Phi)$ is then defined as $M_i(\Phi)$.

It is well known that the evaluation of an attribute grammar takes linear time when sending the evaluation of a semantic rule as one unit of time (see, e.g., [8]). Since a finite propositional formula can be found in evaluated in constant time, the evaluation $M(\Phi)$ of a MMM M on a tree Φ can then be computed in time linear in the size of Φ.

We actually total valuation v of Φ is said to satisfy Φ if $\Phi(x[n][m])$ equals the truth value taken by $\Psi(\Phi, \Phi, n, m)$ when x, the null situation n and make m of Φ null that n is an situation of the total of m.

We shall make use of the following lemma:

Lemma 3.11 The real MPP Φ and two Φ, $\Psi(\Phi)$ is the only valuation that satisfies Φ.

Proof. It follows immediately from the definition that $\Psi(\Phi)$ satisfies Φ.

Suppose that v satisfies Φ. We now show by induction on t that if $\Phi(n[m])$ is defined in $\Phi_t(\Phi)$ then $\Phi_t(\Phi)(n[m]) = v(n[m])$. This clearly holds for $t = 0$. Suppose $t \geq 0$ and $n[m]$ is defined in $\Phi_t(\Phi)$. If $n[m]$ is already defined in $\Phi_{t-1}(\Phi)$ then the claim follows by the inductive hypothesis. If $n[m]$ is not defined in $\Phi_{t-1}(\Phi)$, then, by definition, the value $\Phi_t(\Phi)(n[m])$ equals the truth value of $\Psi(\Phi, \Phi, n, m)$ under the valuation $\Phi_{t-1}(\Phi)$. By assumption $v(n[m])$ equals the truth value of $\Psi(\Phi, \Phi, n, m)$ under the valuation v. By the inductive hypothesis we have that $\Phi_{t-1}(\Phi)(m) = v(m)$, for all m that are defined in $\Phi_{t-1}(\Phi)$. Hence, $\Phi_t(\Phi)(n[m]) = v(n[m])$.

By induction of $\Phi(\Phi)$ the lemma now follows. ∎

A MPP Φ can be used in a simple way to express among (i.e., 1-ary) quantum. Among the situations in the consistency of Φ, we distinguish some situations small, call them:

Definition 3.12 A MPP Φ expresses the among query Φ defined by

$$\Phi(\Phi) = \{x \mid \Phi(\Phi)(\mathrm{small}(x)) = 1\},$$

for every tree Φ.

Example 3.13 Recall the MPP of Figure 1. Φ definition two of the following processes can be viewed naturally as a strategy over the alphabet $\{x, y\}$. Every node labeled Φ in the tree represents a position in the strategy. Now consider the numerical value defining the equivalent situation zero. Play one the existential Defender-$\exists$ has say node x, represented in tree Φ x is zero-numbered when reaching $\exists$ from the Defender. The numerical value defining the inductive situation v_Def one the existential top-Down $v\_\mathrm{Def}(x)$ in tree Φ the letter x occurs in the strategy corresponding Defender position x. Finally, the numerical value for the situation small simply Defender $\mathrm{small}(x)$ as $v\_\mathrm{Def}(x)$ Φ $\mathrm{zero}(x)$. Hence, the MPP expresses the query calculating Hence zero-numbered positions that zero after or x in the strategy. The illustration is given in Figure 3. ∎

A MPP can also be used to express Boolean (i.e., nullary) queries. Among the situations of the start symbol, we distinguish some situations small, call them:

Definition 3.14 A MPP Φ expresses the Boolean query Φ defined by

$$\Phi(\Phi) = \begin{cases} \mathrm{true} & \text{if } \Phi(\Phi)(\mathrm{small}(x)) = 1, \\ \mathrm{false} & \text{otherwise,} \end{cases}$$

for every tree Φ. Here x denotes the root of Φ.

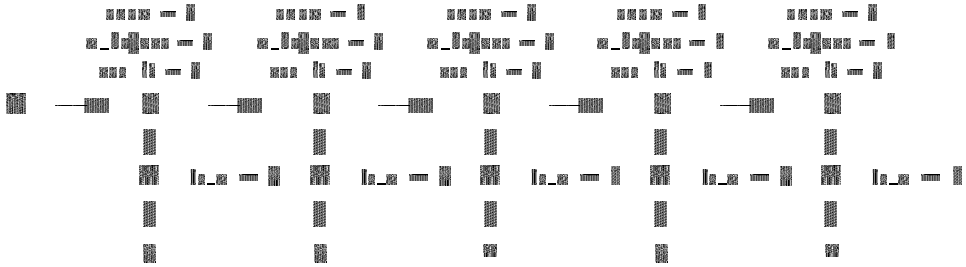

Figure 5: A derivation tree and its relation defined by the DFS of Figure 1.

5.3 Relation-valued attribute grammars for relational queries

In this section, we generalize AGFs to relation-valued attribute grammars (RVAGs). We start by giving an example.

Example 5.10 As a first example, consider the DFS class in Figure 1. A derivation tree of the underlying grammar module is not (G) of Document (D). Each Document is a list (M) of paragraphs (p). The equivalent attribute model of M and G is relation-valued; on any tree, the value of model at the root will be the binary relation containing all triples (D, p, t) such that D is a Document, p is the first paragraph of D, and t is the last paragraph of D. This relation is computed using the equivalent attribution that sets text of M and G, for every Document node n, first(n) contains the first paragraph of that Document, and last(n) contains the last. Other attributes are computed in turn using the inherited attribute tags and the equivalent attribute set of M, which are Boolean-valued; for any M-node n, tags(n) is true if n marks the beginning of a Document, and ends(n) is true if n marks the end. Note that we have are first-order expressions, rather than propositional ones, in defining the values of the attributes. ■

Let us introduce the Hennessy Milner RVAGs and RVAGs more formally.

Definition 5.12 An attribute n we associate an arity a_n (a natural number). A RVAG-rule in the context (p, n, i), with p = $\ldots$ $\ldots$ is an expression of the form

$$n[i] := g[e_1, \ldots, e_{a_n}],$$

where g is a first-order logic formula over the vocabulary

$$\bigcup_{j=1}^{k} \{ f_j[i] \mid 1 \le j \le k \} \cup \{ \mathrm{tag}[i], \mathrm{tag}[i_1], \ldots, \mathrm{tag}[i_k] \},$$

where for each j = 1, $\ldots$, k, $f_j[i]$ is a relation symbol of arity a_j, and tag is a unary symbol. A valuation of a derivation tree T is a function that maps each pair (n, n), where n is a node labelled , and n is an attribute of , to an a_n-ary relation over the nodes of T. A RVAG M consists of an attribute grammar consisting together with a mapping associating to each unary a RVAG-rule in that unary.

```
[redacted] ::= [redacted]
[redacted] ::= [redacted] | [redacted]
[redacted] ::=
[redacted] ::= [redacted]
           [redacted] ::= true
           [redacted] ::= false
[redacted] ::= [redacted] [redacted] [redacted] else [redacted]
           [redacted] ::= [redacted] [redacted] [redacted] else [redacted]
           [redacted] ::= false
           [redacted] ::= false
[redacted] ::= true
           [redacted] ::=
           [redacted] ::=
```

Figure 5: Example of a [redacted].

Definition 3.17 Let [redacted] be a [redacted], and let Σ be a finite set. Let $\mathrm{rel}(i) := \mathrm{R}$ be the [redacted]-rule in the context $[\mathrm{R}, n, i]$. Let m_x be a rule with attributes $\mathrm{m}_1, \ldots, \mathrm{m}_\mathrm{x}$ defined by R. Then the formula obtained from R by replacing each occurrence of a relation symbol $\mathrm{S}(i)$ by the relation symbol $\mathrm{S}(\mathrm{m}_i)$, and by replacing each constant symbol i by the rule m_i, is defined by $\mathrm{W}(\mathrm{R}, \Sigma, n, \mathrm{m}_i)$.

Definition 3.18 Let [redacted] be a [redacted] and Σ a finite set. We define a sequence $(\mathrm{M}_i(\mathrm{m}))_{i \geq 0}$ of partial valuations as follows:

- $\mathrm{M}_0(\mathrm{m})$ is the empty valuation ($\mathrm{M}_0(\mathrm{m})$ is nowhere defined).
- $\mathrm{M}_i(\mathrm{m})$, for $i \geq 1$, is defined as the following extension of $\mathrm{M}_{i-1}(\mathrm{m})$. For every $\mathrm{v}(\mathrm{m})$, if M_{i-1} is defined on all relational symbols that occur in $\mathrm{W}(\mathrm{R}, \Sigma, n, \mathrm{m})$, then $\mathrm{M}_i(\mathrm{m})$ is defined on $\mathrm{v}(\mathrm{m})$ as the relation obtained by evaluating the [redacted]-formula $\mathrm{W}(\mathrm{R}, \Sigma, n, \mathrm{m})$ where each relation symbol $\mathrm{S}(\mathrm{m})$ in $\mathrm{W}(\mathrm{R}, \Sigma, n, \mathrm{m})$ is interpreted by $\mathrm{M}_{i-1}(\mathrm{S}(\mathrm{m}))$.

The valuation $\mathrm{M}(\mathrm{m})$ is then defined as $\mathrm{M}_i(\mathrm{m})$, where i is such that M_i is a total valuation.

By ordinary total valuation a of Σ is said to satisfy R if $\mathrm{v}(\mathrm{v}(\mathrm{m}))$ equals the relation defined by the [redacted]-formula $\mathrm{W}(\mathrm{R}, \Sigma, n, \mathrm{m})$, where each relation symbol $\mathrm{S}(\mathrm{m}_i)$ is interpreted by $\mathrm{a}(\mathrm{S}(\mathrm{m}_i))$. Analogous to Lemma 3.11, one can prove the following lemma:

Lemma 3.1 For each [redacted] R and tree Σ, $\mathrm{M}(\mathrm{m})$ is the only valuation that satisfies R.

A [redacted] can be used to express 0-ary queries in a simple way. Among the attributes of the start symbol R we designate some 0-ary attribute result, and define:

```
[redacted]   [redacted]  :=  [redacted] = [[x, y]] = [[y, y]]
                                 = [redacted] = [[y]]
[redacted]   [redacted]  :=  [[y]] = [redacted] = [redacted]
             [redacted]  :=  [redacted] = [[x]]
                                 = [[redacted] = [redacted]]
                                 = [redacted] = [redacted]
[redacted]   [redacted]  :=  [[y], [x]]
             [redacted]  :=  [[x, y], [y, y], [x, x]]
```

Figura 5: Compilazione di due [redacted] in due [redacted] dalla [redacted].

Definizione 3.24 Il [redacted] [redacted] appunto che quando [redacted] dell'albero: ha una foglia x, [redacted] uguale alla coda di [redacted] in [redacted], allora è la testa così di x.

Esempio 3.25 Quello esempio di è [redacted] la traduzione in Figura 5. Alla linea [redacted], 5-sig] quando incontrato [redacted] [redacted] [redacted] [redacted] in è [redacted] [redacted] di [redacted], corrispondendo la è particolare [redacted] [12] di [redacted]. Alla esempio non [redacted] [redacted] [redacted] [redacted]. ■

5 Regressi e guasti di [redacted]

In [redacted] [redacted] [redacted] [redacted]

0.0.3 Monadic second-order logic

Let τ be the relational subsig of bounded arities in the relational signature Ω. A Ω-tree T can be naturally viewed as a finite structure (in the sense of multisorted logic [8, 9]) over the binary relations $\{R_1, \ldots, R_n\}$ and the unary relations $\{M_p \mid p \in \Omega\}$. We denote the restriction of T to τ by τ_T. The domain of T, viewed as a structure, consists the set of nodes of T. The relations R_i in T consist the set of pairs (x, x') such that x' is the i-th child of x in T. The set M_p in T consists the set of p-labeled nodes of T.

Monadic second-order logic (MSO) allows the use of set variables ranging over sets of nodes of a tree, in addition to the individual variables ranging over the nodes. Formulas in prenexed τ disjoint first-order logic. A detailed exposition on this logic can be found in, e.g., Ebbinghaus and Flum's book [3].

We denote the set of monadic second-order formulas over τ_Ω by MSO(Ω). If Ω is clear from the context we omit it and just write MSO.

Consider a Ω-tree T and a sequence $x_1, \ldots, x_n$ of nodes of nodes of T. We can view the tuple $(T, x_1, \ldots, x_n)$ as a labeling of T: a node x in T is labeled $\{x_1, \ldots x_n\}$, where for $i = 1, \ldots, n$,

$$x_i := \begin{cases} 1 & \text{if } x = x_i, \\ 0 & \text{otherwise.} \end{cases}$$

Let Ω^n be the ranked signature obtained from Ω as follows: for each function symbol f in Ω, we have for each $\bar{\alpha} \in \{0, 1\}^n$, the function symbol $f_{\bar{\alpha}}$ in Ω^n, of the same arity as f. As $|\Omega| \in$ is a Ω-tree, then $(T, x_1, \ldots, x_n)$ is a Ω^n-tree.

Let $\varphi(M_1, \ldots, M_n)$ be an MSO(Ω)-formula where $M_1, \ldots, M_n$ are free set variables. φ defines, the set

$$\mathcal{M}_\varphi := \{(T, x_1, \ldots, x_n) \mid T \models \varphi(x_1, \ldots, x_n)\},$$

are the closed as a set of Ω^n-trees.

Definition 0.3 A Ω^n-tree language $\mathcal{M}$ is MSO(Ω)-definable if there exists an MSO(Ω)-formula $\varphi(M_1, \ldots, M_n)$ such that $\mathcal{M} = \mathcal{M}_\varphi$.

A classical theorem in the theory of logic and tree languages [10] says that:

Theorem 0.0 [8, 10] Let Ω be a ranked alphabet. A Ω^n-tree language is MSO(Ω)-definable if and only if it is recognizable.

An important special case is when the tree language is defined by an MSO(Ω)-sentence φ. Then the tree language defined by φ, $\mathcal{M}_\varphi := \{T \mid T \models \varphi\}$, is a set of Ω-trees, instead of a set of trees over an arbitrary of Ω.

0.0.0 Weak alternating automata

We now define tree automata with Ω-trees. We consider with the parameter Ω a ranked signature Ω_Ω as follows: every terminal symbol of Ω is a terminal symbol in Ω, and every production $p \to \ldots$ of Ω is a function symbol in Ω of arity n. A Boolean tree of Ω can now be naturally viewed as a Ω_Ω-tree.

In the previous section we defined [illegible] over $\mathbb{R}$-trees. However, in the context of Dedekind trees, when we use the algebraic $\mathbb{R}_{\mathbb{R}}$, it is a bit more convenient to use the assembling $\mathbb{R}_{\mathbb{R}} = [\mathbb{R}_1, \ldots, \mathbb{R}_n, \{\mathbb{R}_m\}_{m \in \mathbb{N}}]$. Here $\mathbb{R}_1, \ldots, \mathbb{R}_n$ are defined as before, and for each $\quad \in \mathbb{N}$, $\mathbb{R}_m$ is the among relations that equations which define in the tree are indexed with $\quad$. The assemblages $\mathbb{R}_{\mathbb{R}}$ and $\mathbb{R}_{\mathbb{R}_m}$ are really the defined in terms of each other. Indeed, for each $\quad \in$

$$\mathbb{R}_m(v) \quad = \quad \bigcup\{\mathbb{R}_n(v) \mid v = \quad m \ldots n\},$$

and for each $v = \quad , m \quad , \ldots \quad , n \in \mathbb{N}$

$$\mathbb{R}_n(v_1) \quad = \quad \{\mathbb{R}_1\} \ldots \{\mathbb{R}_n\} \left[\bigcup_{i=1}^{n} \mathbb{R}_i(v_1, v_i) \cup \bigcup_{i=1}^{n} \mathbb{R}_m(v_i) \right].$$

Hence, w.l.o.g., we use the assembling $\mathbb{R}_{\mathbb{R}}$.

In the following, unless explicitly specified otherwise, I use say 'tree' as always meaning 'Dedekind tree of $\mathbb{N}$'.

5.5 Main Theorem

We now show that the among genus expressed by $\mathbb{R}$[illegible] is really Hence Dedekind in $\mathbb{N}$[illegible]. $\mathbb{N}$[illegible] can be used in the usual way to define among genus.

Definition 5.5 Let $g(v)$ be an $\mathbb{N}[illegible]$-formula. Then g defines the among genus $\mathbb{R}$ Dedekind $\mathbb{R}_g$

$$\mathbb{R}(v) := \{w \mid \exists \mid g(w)\},$$

for among tree $\mathbb{R}$.

Theorem 5.5 Among genus expressible by a [illegible] is definable in [illegible].

Proof. Let $\mathbb{R}$ be a [illegible]. We have been Theorem 5.11 that for each tree there exists only one relation that satisfies $\mathbb{R}$. In [illegible] we see really Define this relation. For each situation n we have a set variable $\mathbb{R}_n$. This variable will contain all the nodes for which the situation n in tree is $\mathbb{R}(v)$. In this set, we consider a formula in such connects nodes in the following way. Consider a node $n(i) := g$ of $\mathbb{R}$ in the subset $\{g, n, i\}$ for some predecessor $g = \quad , m \quad , \ldots \quad , n$ of $\mathbb{R}$. Define the formula $\mathbb{R}_{g,n,i}(x_n)$ as

$$\mathbb{R}_{g,n,i}(x_n, \{\mathbb{R}_j\}_{j \in \mathbb{N}}) :=$$

$$\{\mathbb{R}_n\}\{\mathbb{R}_1\} \ldots \{\mathbb{R}_n\} \left[\bigcup_{j=1}^{n} \mathbb{R}_{n_j}(x_j) \cup \bigcup_{j=1}^{n} \mathbb{R}_j(x_1, x_j) \cup x_n = x_i \cup \bar{g} \right],$$

where $\bar{g}$ is obtained from g by replacing each propositional symbol $\{j\}$ occurring in g by $\mathbb{R}_j(x_j)$. Intuitively, formula $(*)$ states that

$$x_1 \quad \ldots \quad x_n$$

is defined by the predicates p. Formals $\bar{x}$ stands that p takes two $x_1, x_2, \ldots, x_n$, i.e., that $n(x_1)$ is true. We now define p_n as the following disjunction over all value taking the attribute n:

$$p_n(x_n, \{M_1(v_1)\}) := \bigvee\{\{p_{n,v_1}(x_n, \{M_1(v_1)\}) \mid \{v_1, n, 1\} \text{ is a valuation}\}.$$

Define $\{\{M_n(v_n)\}$ as the formula

$$\bigvee_{v_n \in V} \{\{x \mid M_n(x) = p_n(x, \{M_1(v_1)\}\}.$$

But if Φ is a true call for call n in M let x_n be a set of values of Φ such that $\Phi \models \{\{x_n(v_n)\}$. Then define the valuation v as follows:

$$v(n)(x) := \begin{cases} 1 & \text{if } x \in x_n, \\ 0 & \text{otherwise.} \end{cases}$$

It follows from the definition of Φ that v satisfies M. Since, according to Lemma 5.11, there exists only one valuation that satisfies M, it follows that for each Φ there exists only one sequence of sets $\{x_n(v_n)\}$ such that $\Phi \models \{\{x_n(v_n)\}$. Hence, the following formula defines the group represented by M:

$$\alpha(x) := \{M_n(v_n) \mid \{\{M_n(v_n)\} \in M_{\ldots}, v(x)\}.$$

■

We now state well known and that make sense.

Theorem 5.2 A group group is expressible by a MMS if and only if it is definable in SSS.

Proof. The only-if direction is given by Lemma 5.1.

For the other direction, consider the group defined by the MMS-formula $\alpha(x)$ with one free individual variable x. Define the MMS-formula $\alpha'(M)$, having one free set variable M that is free individual variables, as $\alpha'(M) := \{M(\{\}) \mid \alpha(x)\}$. The formula $\alpha'(M)$ defines M as the set of values that are selected by $\alpha(x)$. By Theorem 5.5, the set $M_\alpha' = \{\{x, x\} \mid \Phi \models \alpha'(x)\}$ is a recognizable M_α^1-tree language. Let $M = \{\ , M, 0\}$ be a tree automaton over M_α^1 recognizing this language. In the proof Definition tree Φ, M accepts exactly those pairs $\{\Phi, x\}$ for which $\Phi \models \alpha'(x)$. Note that we have constructed α' in such a way that for every tree Φ there is only one set x of values of Φ for which $\Phi \models \alpha'(x)$, namely $x = \{x \mid \Phi \models \alpha(x)\}$. In other words, for each Φ, the automaton M will accept only one M-Φ-labeling of Φ, and this "accepted labeling" labels with 1 precisely those nodes in the result set Φ of the group defined by α.

Then. The theorem is proved if we can construct a MMS Φ with an attribute result, such that for each tree Φ and each value v of Φ, evaluating the tree in $\{\Phi\}$ M is labeled Φ in the labeling of accepted by M. This can be achieved by simulating the transition of M on all possible M-Φ-labelings of Φ. $\{\Phi\}$ is impossible to realize

[illegible] [illegible] [illegible] [illegible], [illegible] [illegible] [illegible]

[illegible] in the line and all the real.[illegible] In the [illegible], lap-lines, pass will about at the real.

Let $j = 1, \ldots, n$, and $\quad$ will the rule

$$\mathrm{real}_{-}\{j\} := \bigcap_{i=1}^{n} \left\{ \mathrm{real}_{-}'\{i\} \mid \mathrm{con}_{-}\{i\}[j] \right\}$$

$$', \; 1, \ldots, n \; ,$$

In $\{0, 1\}$ and that

$$\{(w, \; 1, \; 1, \ldots, \; n) = ' \mid j = \}$$

- With the rule

$$\mathrm{real}\{j\} := \bigcap_{i=1}^{n} \left\{ \mathrm{real}_{-}\{i\} \mid \mathrm{con}_{-}\{i\}[j] \right\}$$

$$, \; 1, \ldots, n \; , \; \text{and}$$

$$\{(w, \; 1, \; 1, \ldots, \; n) = \},$$

and and for every j and that $\quad$ is a terminal set

$$\mathrm{real}'\{j\} := \bigcap \left\{ \mathrm{real}_{-}\{i\} \mid \quad , \{i\} \; j\{i\} = \right\}.$$

Proposition. We now establish the correctness of $\mathcal{R}$. Let π be a derivation tree Δ. Let Γ be a labelling of the nodes of Δ with Γ as Δ. We denote the corresponding labelled tree by Δ_{Γ}. If m is a node of Δ, then denote the subtree of Δ with root m by $\Delta\{m\}$. If Γ' is a labelling of $\Delta\{m\}$ such Γ is a labelling of Δ, then $[\Gamma|\Gamma']$ is the following labelling of Δ; for each node m of Δ,

$$[\Gamma|\Gamma'](m) = \begin{cases} \Gamma'(m) & \text{if } m \text{ is a node of } \Delta\{m\}, \\ \Gamma(m) & \text{otherwise.} \end{cases}$$

The following lemma can be proved by a straightforward induction on the height of m.

Lemma 2. For every node m of Δ, $\mathrm{con}_{-}\{m\}$ is true if and only if there exists a labelling Γ of Δ such that Δ_{Γ} satisfies ϕ at m in the semantics on input $\Delta\{m\}$.

Since Δ_{Γ} is deterministic, if for a node m, both $\mathrm{con}_{-}\{m\}$ and $\mathrm{con}_{-}\{m\}$ are true and $\quad \neq \quad$, then there exist two different labellings Γ_1 and Γ_2 for $\Delta\{m\}$ such Δ_{Γ_1} satisfies ϕ at m in the semantics on $\Gamma_1[\Delta\{m\}]$ $[\Gamma_2[\Delta\{m\}]]$. Then in merely one labelling Γ'' of Δ such that satisfies $\Gamma''\{m\}$. In particular, satisfies ϕ at the root r of Δ as $\Gamma''\{m\}$. Hence, then in merely one $\quad$ satisfies $\mathrm{con}_{-}\{m\}$ is true. Hence, by definition of the semantics rules for the attribute $\mathrm{real}_{-}\{m\}$, only one $\mathrm{real}_{-}\{m\}$ is true.

[illegible] got off at the [illegible] by allowing the dot symbol to be a lookahead attribute. The [illegible] of attribute grammars [illegible] such [illegible] clause, [illegible]. [illegible] [illegible] [illegible] the dot symbol.

Lemma 6.3 [illegible] $\mathrm{rank}(m)$ is less than $\mathrm{rank}(\mathrm{Ext}^1(m,m))$ [illegible] m [illegible] $\mathrm{Ext}^1(m)$.

Proof. We show this by induction on the length of m. The base case, where m is $\mathbb{k}$, has just been treated.

Now, consider a cycle $m_\bullet$ with simples $m_1, \ldots, m_n$ indexed by the partition $\mathbb{N} = \mathbb{k} \cup \mathbb{l} \cup \ldots \cup \mathbb{n}$. Put $\mathbb{j} \in \{\mathbb{1}, \ldots, \mathbb{n}\}$.

(i) Suppose $\mathrm{rank}(m_\mathbb{j})$ is large. Since there exists $\mathbb{i} \in \{\mathbb{0}, \mathbb{1}\}$, such $\mathbb{i}_1, \ldots, \mathbb{i}_n \in$ with $\mathbb{i}_\mathbb{j} = \mathbb{i}$, such that

$$\mathrm{Ext}(m, \mathbb{i}_1, \mathbb{i}_2, \ldots, \mathbb{i}_n) = \mathbb{0}, \qquad (\mathbb{0})$$

and

$$\mathrm{rank}(\mathbb{i}_\mathbb{j}) \le \sum_{\mathbb{i}=\mathbb{1}}^{\mathbb{n}} \mathrm{rank}(\mathbb{i}_\mathbb{i})$$

is large. Since $\sum_{\mathbb{i}=\mathbb{1}}^{\mathbb{n}} \mathrm{rank}(\mathbb{i}_\mathbb{i})$ is bounded above that has rank $\mathbb{1} = \mathbb{1}, \ldots, \mathbb{n}$ there exists a indecomposable $\mathbb{i}_\mathbb{i}$ of $\mathrm{Ext}(\mathbb{i}_\mathbb{i})$, such that [illegible] $\mathbb{i}_\mathbb{i}$ of $\mathrm{Ext}(\mathbb{i}_\mathbb{i})$ [illegible] $\mathrm{Ext}(\mathbb{i}_\mathbb{i})$ be bounded: $\mathrm{Ext}(\mathbb{i}_\mathbb{i}) = \mathbb{0}$ [illegible] $\mathbb{i} \in \mathrm{Ext}(\mathbb{i}_\mathbb{i})$ [illegible] $\mathrm{Ext}(\mathbb{i}) = \mathbb{0}$. Since (0) is bounded above [illegible] $\mathbb{i}$ of $m_\mathbb{i}$ [illegible] $\mathrm{Ext}(\mathbb{i}_\mathbb{i})$. Since $\mathrm{rank}(\mathbb{i}_\mathbb{i})$ is bounded by the indecomposable dimension above [illegible] $\mathbb{i}$ of $m_\mathbb{i}$ [illegible] $\mathrm{Ext}^1(\mathbb{i})$. Hence, the tree $\mathrm{Ext}^1(\mathbb{i})(\mathbb{i})$ is supported by [illegible]. Since there is only one supported indecomposable, it follows that [illegible] $\mathbb{i}$ of $m_\mathbb{i}$.

(ii) Suppose [illegible] $\mathbb{i}$ of $m_\mathbb{i}$ [illegible] $\mathrm{Ext}^1(\mathbb{i})$. Let $\mathbb{i}'$ be the class such that [illegible] $\mathbb{i}$ of $m_\mathbb{i}$ [illegible] $\mathrm{Ext}^1(\mathbb{i})$. Since there is the $\mathbb{i}_1, \ldots, \mathbb{i}_\mathbb{n}$ [illegible], with $\mathbb{i}_\mathbb{j} = \mathbb{i}$, such $\mathbb{i} \in \{\mathbb{0}, \mathbb{1}\}$ such that $\mathrm{Ext}(m, \mathbb{i}_1, \ldots, \mathbb{i}_n) = \mathbb{0}$, such such that [illegible] $\mathbb{i}$ of $m_\mathbb{i}$ [illegible] $\mathrm{Ext}^1(\mathbb{i})$, for $\mathbb{i} = \mathbb{1}, \ldots, \mathbb{n}$. By the indecomposable dimension $\mathrm{rank}(\mathbb{i}'_\mathbb{i})$ is large, and by Lemma 6.2, $\mathrm{rank}(\mathbb{i}_\mathbb{i})$ is large for such $\mathbb{i} = \mathbb{1}, \ldots, \mathbb{n}$. Hence, $\mathrm{rank}(m_\mathbb{j})$ is large by induction. ∎

We now show that for such cycle m, $\mathrm{rank}(\mathbb{j}(m))$ is large in $\mathbb{j}(m)$ if such only if $\mathrm{Ext}^1(m) = \mathbb{0}$. Since for every indecomposable tree $\mathbb{0}$, $\mathrm{Ext}^1(m) = \mathbb{0}$ exactly if $\mathbb{0} \models g(m)$, the proof is complete.

1. Put m to be a local cycle indexed with $\mathbb{0}$.

   (i) Suppose $\mathrm{rank}(\mathbb{j}(m))$ is large. Hence, there exists a class $\mathbb{0}$ such that $\mathrm{rank}(\mathbb{j}(\mathbb{0}))$ is large and $\mathbb{j}(\mathbb{0}) = \mathbb{0}$. It follows from Lemma 6.2, that [illegible] $\mathbb{0}$ of m in $\mathrm{Ext}^1(\mathbb{j})$. It remains to the case that $\mathbb{j}(\mathbb{0}) = \mathbb{0}$, however there [illegible] small simple two indecomposable of $\mathbb{0}$. Hence, $\mathrm{Ext}^1(m) = \mathbb{0}$.

   (ii) Suppose $\mathrm{Ext}^1(m) = \mathbb{0}$. Put $\mathbb{0}$ to the class such that [illegible] $\mathbb{0}$ of m in $\mathrm{Ext}^1(\mathbb{j})$. Since $\mathbb{j}(\mathbb{0}) = \mathbb{0}$. Hence, $\mathrm{rank}(\mathbb{j}(m))$ is large.

2. Put m to be an indecomposable cycle, with simples $m_1, \ldots, m_n$, indexed by partition $\mathbb{N}$.

(i) Suppose $\mathrm{multi}(\alpha)$ is true. Hence, there exist elements $\alpha_1, \ldots, \alpha_n$ such that $\mathrm{Sig}(\mathbb{T}, \alpha_1, \ldots, \alpha_n) = \Sigma$, such that $\mathrm{ind}(\alpha) \in \mathbb{S}_{\Sigma}$, that $\mathrm{ind}(\alpha_i)$ is true. By Lemma 5.3 that $\mathbb{T}$ contains elements α of α, such elements α_i of α_i in $\mathbb{T}^{\alpha}(\mathbb{S})$, for $i = 1, \ldots, n$. It remains to show that $\mathrm{Sig}(\mathbb{T}, \alpha_1, \ldots, \alpha_n) = \Sigma$. However there $\mathbb{T}$ must exist by hypothesis of 2. Hence, $\mathbb{T}^{\alpha}(\alpha) = 1$.

(ii) Suppose $\mathbb{T}^{\alpha}(\alpha) = 1$. Let α be the elements that $\mathbb{T}$ contains of α in $\mathbb{T}^{\alpha}(\mathbb{S})$, such for $i = 1, \ldots, n$, let α_i be the elements that $\mathbb{T}$ contains of α_i in $\mathbb{T}^{\alpha}(\mathbb{S})$. Then $\mathrm{Sig}(\mathbb{T}, \alpha_1, \ldots, \alpha_n) = \Sigma$. Hence, $\mathrm{multi}(\alpha)$ is true. ■

We conclude about proof of Theorem 5.4 we obtain a normal form for $\mathbb{T}\mathbb{S}$. The $\mathbb{T}\mathbb{S}$ Resolution is the proof is special in two ways. First, it works only positive formulas (excluding only the connectives $\neg$ and $\bot$, without $\neg$) in the consequents rules. Second, it can be understood as way from $\mathbb{G}$ into Gallaire-up proof followed $\mathbb{G}$ into top-down proof. So we have the following:

Corollary 6. Every $\mathbb{T}\mathbb{S}$ is equivalent to one which uses only positive formulas in the consequents rules, and consequently which can be understood in two proof procedures, which is clausal-$\mathbb{S}$-proof [8].

Actually, proof of the above corollary, that uses our change that we equivalent $\mathbb{T}\mathbb{S}$ which uses only positive rules, can also quite easily be done directly. But $\mathbb{T}$ be $\mathbb{T}\mathbb{S}$ into the attributes grammar corresponding $(\mathbb{S}, \mathrm{Syn}, \mathrm{Inh}, \mathrm{Out})$. We construct an equivalent $\mathbb{T}\mathbb{S}$ $\mathbb{T}'$ into the attributes grammar corresponding $(\mathbb{S}', \mathrm{Syn}', \mathrm{Inh}', \mathrm{Out}')$ that have not one negative in the consequents rules in the following way. For each attribute a in Out or attribute $\mathbb{S}n$ that occurs into the attribute a in the rules. Formally, $\mathbb{S}' = \mathbb{S} \cup \{\mathbb{S}n \mid n \in \mathbb{S}\}$, for each grammar symbol $\ldots$,

$$\mathrm{Syn}'(\ldots) = \mathrm{Syn}(\ldots) \cup \{\mathbb{S}n \mid n \in \mathrm{Syn}(\ldots)\},$$

and

$$\mathrm{Inh}'(\ldots) = \mathrm{Inh}(\ldots) \cup \{\mathbb{S}n \mid n \in \mathrm{Inh}(\ldots)\}.$$

For each rule $r(i) := g$ of $\mathbb{T}$ in context $(\mathbb{S}, i, r)$, add the rule $r(i) := \bar{g}$ in context $(\mathbb{S}, r, i)$ and the rule $\mathbb{S}r(i) := \overline{\neg g}$ in context $(\mathbb{S}, \mathbb{S}r, i)$ in $\mathbb{T}'$. The formula $\bar{\mathbb{S}}$, where $\mathbb{S} = g$ or $\mathbb{S} = \neg g$, is obtained from $\mathbb{S}$ by transforming it into disjunctive normal form and then replacing each literal $\neg l(\ldots)$ by $\mathbb{S}l(\ldots)$.

Example 6.10 The $\mathbb{T}\mathbb{S}$ in Example 5.5 now replace in order all rules as we have mentioned previous. In Figure 5 is $\mathbb{T}\mathbb{S}$ in Datalog that calculate those rules without using negation. For clarity we replace the attributes $\mathbb{S}\mathrm{ccc}$ by all. ■

6.5 Gallaire-up as only for Horizon $\mathbb{T}\mathbb{S}$ parser

Gallaire also of $\mathbb{T}\mathbb{S}$ is that of a top-down parser of Datalog Gallaire-up for semi-formula, although in the more classical top-down parsing algorithms provided by Earley [10]. The top-down parsing is provided by the two Dif-ferent types of attributes in a $\mathbb{T}\mathbb{S}$: inherited, equivalent attributes provide the Gallaire-up direction, and inherited attributes provide the top-down direction.

The following propositions relate $\mathbb{T}\mathbb{S}$s and two schemas.

```
M := 0     w_lower[0] := false
M := M+1   w_lower[M] := to_w[0] ∧ w_lower[M]
           ssss[M] := aaa[M]
           aaa[M] := ssss[M]
           ssssb[M] := ssss[M] ∨ w_lower[M]
M := M     ssss[M] := false
           aaa[M] := true
           ssssb[M] := false
M := w     to_w[M] := true
M := p     to_w[M] := false
```

Figure 1: Example of a DFA without negation.

Proposition 0.11 The only two automata DFA that satisfy a DFA M and that for some Boolean b, DFA example 1 M and only M M example 1. This DFA are only equivalent situation.

Proof. The transition of DFA $= \{ \ , M, 0\}$ as 1 are really the standard by a DFA M having equivalent situation $\ $ has all states $\ $ to $\ $. If w is a suffix of 1, then the situation value $\ [w]$ is true to $M[w]$ $\ $ DFA accepts state $\ $ if w is the transition on 1. But $p = \ _p \ = \ _1, \ldots, \ _p$ is a prefixes of M, $M[w] := \{ \ \in \{1, \ldots, w\} \mid \ _p$ is a terminal$\}$, and $\ [w] := \{1, \ldots, w\} - M[w]$. DFA has only $\ $ the accepts rule

$$[w] := \bigcup_{\substack{\ \in \{1, \ldots, \ _p\} \\ \text{and has only } 1 \in M[w],\ 0 \in \ _p = \ _1\}}} \ [\ ].$$

Finally, the situation accept of M is defined by the rule $accept[w] := \bigcup_{\ \in \ } \ [w]$. ■

It was follows from Theorem 0.9 and Proposition 0.11, that every DFA-definable Boolean query is expressible by a equivalent DFA. This then leads to the following Datalog-as property of Boolean DFA queries:

Corollary 0.12 For every DFA M there is a DFA M' having only equivalent situation, such that M and M' express the same Boolean query.

In the general case of multiway situation programs, whose accepts value are the multiway acceptable Boolean, it is well known that the non-deterministic situation are the standard using equivalent situation only [10]; so there are that a similar phenomenon holds when accepts value are only the propositional formulas.

Corollary 0.12 does not hold for DFA expressing query queries, as illustrated in the following example.

Example 0.13 Consider again the program in Example 0.8. The query selling has all sales that occur as an even position when counting top-down in an expressible by a DFA with only equivalent situation; similarly, the analogous query that counting bottom-up in an expressible with only bounded situation.

[redacted] [redacted] [redacted] [

We now define local Ehrenfeucht logic [illegible], in the same way as FMT except that we call formulas of the form $\mathrm{FMT}[q, m](x_1, \ldots, x_k)$, q free in the position.

Note that our definitions of FMT and MSO differ from those in the literature: we do not allow nesting of Ehrenfeucht games or do not allow permutations in the formula constituting the Ehrenfeucht. However, these differences are do disappear still, our definitions are equivalent to the usual ones [illegible].

6.1.3 Bisimulation of finite Kripke

Consider a FMT-formula of the form $\mathrm{FMT}[q, m](x_1, \ldots, x_k)$. If there exist natural numbers r and D such that for every tree t, q consider the partial Ehrenfeucht game of at most $r \cdot |t|$ m D turns, where $|t|$ denotes the number of nodes in t, then we say that q is locally invariant by the partial Ehrenfeucht accounting. We define the logic FMT-BISM as the fragment of FMT where only partial Ehrenfeucht formulas are allowed.

FMT-BISM is then the fragment of FMT-BISM that only allows formulas where the Ehrenfeucht operator that are both positive and locally invariant.

6.1.4 Simultaneous Ehrenfeucht logic

Let $\varphi_1(\bar{x}_1, \bar{Y}_1, \ldots, \bar{Y}_k), \ldots, \varphi_n(\bar{x}_n, \bar{Y}_1, \ldots, \bar{Y}_k)$ be n equations of first-order formulas, where for $i = 1, \ldots, n$, $\bar{Y}_i$ is $|\bar{x}_i|$-ary relation variable. We n time t, consider for $i = 1, \ldots, n$, the stages defined by

$$\varphi_i^0(t) := t,$$
$$\varphi_i^{m+1}(t) := \{(x_1, \ldots, x_k) \mid t \models \varphi_i(x_1, \ldots, x_k, \varphi_1^m(t), \ldots, \varphi_n^m(t))\}.$$

We say that the equations consider a simultaneous Ehrenfeucht game at t if there exists an m such that for all $i = 1, \ldots, n$, $\varphi_i^m(t) = \varphi_i^{m+1}(t)$. We denote the relation defined by φ_i in the Ehrenfeucht by $\varphi_i^\infty(t)$. If there does not exist a simultaneous Ehrenfeucht at t, then $\varphi_i^\infty(t)$ is defined as the empty set. We now define simultaneous partial Ehrenfeucht logic [illegible] as follows: formulas are constructed just as in first-order logic, with the additions that we also allow formulas of the form $k\text{-}\mathrm{MFT}_i[\varphi_1, \ldots, \varphi_n, \bar{Y}_1, \ldots, \bar{Y}_k](x_1, \ldots, x_k)$, where $\bar{Y}_j$ is s-ary and φ_i is a first-order formula for $i = 1, \ldots, n$. The semantics is defined as follows: for any tree t, and values $x_1, \ldots, x_k$

$$t \models k\text{-}\mathrm{MFT}_i[\bar{\varphi}, \bar{\bar{Y}}](x_1, \ldots, x_k) \iff (x_1, \ldots, x_k) \in \varphi_i^\infty(t).$$

We say that the equations of first-order formulas $\varphi_1, \ldots, \varphi_n$ is locally invariant by the simultaneous partial Ehrenfeucht accounting if there exist natural numbers r and D such that for every finite tree t, the equations of first-order formulas $\varphi_1, \ldots, \varphi_n$ consider the simultaneous partial Ehrenfeucht after at most $r \cdot |t|$ m D turns. We define the logic k-MFT-BISM as the fragment of k-MFT where only simultaneous partial Ehrenfeucht with equations of first-order formulas are allowed.

The next proposition states that k-MFT-BISM is equivalent to FMT-BISM. The proof is mostly an the proof of the simultaneous inductive Dawes Dawes game the theory of inductive definitions and finite model theory [illegible, illegible].

Proposition 6.1 Every $\mathbb{R}$-[illegible]-[illegible]-formula of the form

$$\mathbb{R}\text{-}\mathrm{[illegible]}_{\ell}(\bar{x}, \bar{\mathbb{M}})(\bar{a}),$$

where $\ell \in \{1, \ldots, k\}$, is equivalent to a [illegible]-[illegible]-formula of the form

$$(\exists \bar{z})(\mathrm{[illegible]}(\mathbb{M}, \mathbb{M})(\bar{a}\bar{z}))_{\ell}$$

6.3 Main Theorem

We now relate [illegible] to [illegible]-[illegible]. [illegible]-[illegible]-formulas are exactly $\mathbb{R}$-ary queries in the following way:

Definition 6.2 Let $g(x_1, \ldots, x_k)$ be a [illegible]-[illegible]-formula. Then g represents the $\mathbb{R}$-ary query $\mathbb{Q}$ defined by

$$\mathbb{Q}(\mathbb{A}) := \{(x_1, \ldots, x_k) \mid \mathbb{A} \models g(x_1, \ldots, x_k)\},$$

for every form $\mathbb{A}$.

Theorem 6.3 A query is expressible by a [illegible] if and only if it is definable in [illegible]-[illegible].

Proof. Only if. Let $\mathbb{M}$ be a [illegible]. We assume w.l.o.g. that no constant symbol appears in exactly all $\{x_0\}_{0 \le i \le n}$, $x_1, x_2, x_3, \ldots$. We define an $\mathbb{R}$-[illegible]-[illegible]-formula that simulates $\mathbb{M}$. We inductively construct all the systems we have as $\{x_0 \mathbb{M} i\}$-ary relations constructs $\mathbb{M}_n$ for each situation n; $\mathbb{M}_n$ stands for the set of all tuples $\{x, x_1, \ldots, x_{n-1}\}$, where x is a node inducted [illegible] such that $n \le \mathrm{[illegible]}$ [illegible], and $\{x_1, \ldots, x_{n-1}\}$ is a tuple in the currently computed value of $\mathrm{[illegible]}(\{n\}\{x\})$. For each situation n there is a formula $\varphi_n(x_0, x_1, \ldots, x_{n-1}, \{\mathbb{M}_i\}_{i \le n})$, defining the new value of $\mathbb{M}_n$ from the old values of the $\mathbb{M}_i$'s, built up as follows. Consider a rule $n(t) := g$ of $\mathbb{M}$ in the context $\{w, n, t\}$ for some predicates $w = \ldots \wedge \mathrm{[illegible]} \wedge \ldots \wedge$ of $\mathbb{M}$. The formula $\varphi_{g,n,t}(x_0, x_1, \ldots, x_{n-1}, \{\mathbb{M}_i\}_{i \le n})$ is defined as

$$(\mathrm{[illegible]}_t)(\mathrm{[illegible]}_1) \ldots (\mathrm{[illegible]}_k) \left(\bigwedge_{i=1}^{k} \varphi_{n_i}(x_i) \wedge \bigwedge_{i=1}^{k} \varphi_i(x_i, x_i) \wedge x_0 = x_1 \wedge \bar{\psi} \right),$$

where $\bar{\psi}$ is obtained from g by replacing each occurrence of $\{\{\}\}(\bar{\alpha})$ by $\mathbb{M}_i(x_i, \bar{\alpha})$, and by replacing each occurrence of the constant symbol $\mathrm{[illegible]}$ by x_1. The formula φ_n then is the disjunction over all rules defining the situation n:

$$\varphi_n(x_0, x_1 \ldots, x_{n-1}, \{\mathbb{M}_i\}_{i \le n}) :=$$
$$\bigvee \{\varphi_{g,n,t}(x_0, x_1 \ldots, x_{n-1}, \{\mathbb{M}_i\}_{i \le n}) \mid (w, n, t) \text{ is a rule}\}.$$

Let $\varphi(\bar{x})$ be the formula

$$(\exists z)\left(\mathbb{M}_m(z) \wedge \mathbb{R}\text{-}\mathrm{[illegible]}_1(x_{m-1}, z, \{x_i\}_{i \le m-2}, \{\mathbb{M}_i\}_{i \le m})(z, \bar{x})\right).$$

[redacted] [redacted] [redacted] [

[redacted] [redacted] [redacted] [

[illegible]. For each $j = 1, \ldots, n$, we have

$$\mathbb{E}[\Omega] = \mathbb{E}[\bar{\omega} \mid \mathbb{E}_j^{\omega}[\bar{\omega}]],$$

where $\mathbb{E}_j^{\omega}$ is obtained from $\mathbb{E}^{\omega}$ by replacing each assumption of $\mathbb{E}_m[\omega]$ as $\mathbb{E}_j[\omega_1, \omega_2]$ by $\mathbb{E}_m[\Omega][\omega]$ as $\mathbb{E}_j[\Omega][\omega_1, \omega_2]$ respectively, and by replacing each assumption of $\mathbb{E}[\Omega]$ by $\mathbb{E}'[\Omega - 1][\Omega]$.

[illegible]. Finally, define

$$\mathbb{E}'[\omega] = \mathbb{E}'[\omega].$$

Note that as every term [illegible], the evaluation of $\mathbb{E}^{\omega}$ performs exactly $5 \cdot |\omega|$ iterations. In each iteration exactly one attribute $\mathbb{E}$ or exactly one attribute $\mathbb{E}'$ is defined. For a term [illegible], let for $1 \leq i$, $\omega_i[\omega]$ $[\mathbb{E}_i[\omega]]$ be the number of $\mathbb{E}$ $[\mathbb{E}']$ attributes that are defined in $\mathbb{E}_i[\omega]$. The correctness of the construction now follows from the following lemma:

Lemma 6.5 *For Ω in a term, let ω be a node of Ω and let $n \in [\mathbb{E}, \mathbb{E}']$. If $n[\omega]$ is defined in $\mathbb{E}_i^{\omega}[\omega]$ but not in $\mathbb{E}_{i-1}^{\omega}[\omega]$, then*

$$\mathbb{E}_i^{\omega}[\omega][n[\omega]] = [\bar{\omega} \mid \omega \models \mathbb{E}^{\omega \cdot \omega_i[\omega]}[\bar{\omega}, \Omega]].$$

The lemma can be proved by induction on the pair $(\omega_i[\omega], \mathbb{E}_i[\omega])$.

Hence, $\mathbb{E}_n[\omega][\mathbb{E}'[\omega]]$ equals the value defined by $\mathbb{EEE}[\mathbb{E}, \mathbb{E}][\bar{\omega}]$, where ω is the root of Ω. ∎

The logic EEE-DIS has a rather obvious equivalence, as it allows the formulas of a formula only when that formula is literally defined, which is not an obvious equivalent property. Actually, we do not know whether these formulations of fixed-point formulas are decidable since of course that problem is undecidable. Even proofs the property can be shown undecidable by a reduction from validity, that are decidable from equivalently Π-form, for some recursive signature Π, fixed-point logic (more precisely recursive-free logic) is decidable [8, 22, 23].

The problem of finite equivalence can be resolved, however, by defining EEE-DIS in an alternative manner. Define this alternative, the formula of any formula is allowed (as that the equivalence is more intuitively decidable). We then hold into the consequence that the formula is performed exactly ω times, where ω is the cardinality of the formula. It is not difficult to adapt the proof of Theorem 6.3 for this alternative view of EEE-DIS.

6.3 Some facts of EEEE

Immerman [10] showed that EEE-DIS captures the complexity class EEEEE$[\omega]$ consisting of all queries computable in time $\mathbb{E}[\omega]$ by a parallel machine with polynomially many processors.

Theorem 6.3 [10] EEEE-EEE = EEEEE$[\omega]$ on the class of all ordered finite structures.

Hence, to show that [illegible] — [illegible] is coNP-hard, we have to prove that (i) and (ii) are coNP-hard, or that (i) and (ii) [illegible] are in NP. The complements of (i) and (ii) are shown to be in NP in Lemma 5.7. Hence, by Theorem 5.6, we have:

Corollary 5.8 It is easy to recognize by a [illegible] if and only if it is compatible to [illegible] parallel time with polynomially many processors.

Lemma 5.7 (i) [illegible] is a [illegible]-[illegible]-formula that uniquely defines a total order on all [illegible].

(ii) [illegible] — [illegible] is the class of all [illegible].

Proof. (i) Example 5.11 shows how an ordering of a strong [illegible] can be obtained using a [illegible]. It is straightforward to generalize this construction to arbitrary finite [illegible]. By Theorem 5.6, this [illegible] is equivalent to a [illegible]-formula.

(ii) In Example 5.11, we saw how we can compute an ordering of a [illegible] using a [illegible]. We can also compute this ordering directly in [illegible]. Hence, the equivalence of [illegible] and [illegible] is [illegible] equivalence in ordinary [illegible]. The proof of the latter is similar to the proof of the lemma that shows that [illegible] equals [illegible] in ordinary structures, [8, Thm 5.5.15] [see also [9]]. Here [illegible] denotes the fragment of [illegible], where every formula is restricted to a polynomial number of literals. ■

5.5 The Ordinary-[illegible] [illegible] only for [illegible]

In this section we prove that equivalent [illegible], i.e., [illegible] that only are equivalent structures, are closely [illegible] [illegible] that are not [illegible] equivalent and [illegible] structures.

For the rest of this section, let [illegible] be the [illegible] [illegible] [illegible]. Let equal_colors be the group that is true on [illegible] when the left colors [illegible] the same number of colors as the right colors. We show that this group cannot be expressed by a equivalent [illegible]. However, it can be expressed by a [illegible].

Proposition 5.9 The group equal_colors is expressible by a [illegible].

Proof. By Theorem 5.5, it suffices to show that equal_colors is definable in [illegible]:

$$g := \{(w)\,|\,(g)\,|\,w \models \varphi \wedge \mathrm{Ex}_g(x) \wedge \mathrm{Ex}_g(y) \wedge \mathrm{Ex}[x, \quad](x, y)\},$$

where

$$\varphi(x, y, \quad) := \{(w)\,|\,(\mathrm{Ex}_g(x) \wedge \mathrm{Ex}_g(x, w) \wedge \mathrm{Ex}_g(x, y)\} \\ \wedge \{(w')\,|\,(\mathrm{Ex}_g(x')) \quad (x', w') \wedge \mathrm{Ex}_g(x', w) \wedge \mathrm{Ex}_g(x', y)\}.$$

This formula evaluates a strong coloring [illegible]. In the first [illegible] of x, the first node of the left colors and the first node of the right colors are put in [illegible]. In the

following formulas, the set pair of corresponding sets is still in , provided it exists. Hence, a literal of sort $|\Phi|$ since as a tree Φ, will also belong to [illegible]-[illegible]. The formula φ also because from it both the left side of the left sections and the left side of the right sections belong to . ■

In the end of this section we prove that equal_sections cannot be expressed by a equivalent [illegible].

Definition 8. A simple [illegible] is a equivalent [illegible] over the signature containing just two only the signature σ has R and only the signature must be R.

The here signature as simple [illegible], and also into that any equivalent [illegible] can be transformed into an equivalent simple one.

For any integers x_1 and x_2 greater than 1, let $\Phi[x_1, x_2]$ denote the tree which has a left section of length x_1 and a right section of length x_2. Not [illegible] is a simple [illegible]. In Lemma 8.11 we show that the values of $\mathrm{Min}[\Phi[x_1, x_2]]|\Phi[x_1, x_2]|$ and $\mathrm{Min}[\Phi[x_1, x_2]]|\Phi[x_1, x_2]|$, where x_1 is the first still and x_2 is the second still of the pair, can be arbitrarily defined in [illegible] over a structure that, essentially, only contains an ordering of part of the formula of $\Phi[x_1, x_2]$. Hence, we call some definitions.

Definition 8.10 Let $x_{\mathrm{M}} = [\mathrm{M}, \mathrm{S}, \mathrm{S}, m]$ be the assertions consisting of the sentinel symbols M, S and S, and the binary relation symbol m. Let x_1 and x_2 be two integers greater than 1.

1. Define $\mathrm{Ch}_1[x_1, x_2]$ as the x_{M}-structure with formula $[1, \ldots, x_1 m x_2 m 1]$, where $\mathrm{M} = x_1 m x_2 m 1$, $\mathrm{S} = x_1$, $\mathrm{S} = x_1 m x_2$, and where m is interpreted as the total order on $[1, \ldots, x_1]$;

2. Define $\mathrm{Ch}_2[x_1, x_2]$ similarly as $\mathrm{Ch}_1[x_1, x_2]$, except that now m is interpreted as the total order on $[x_1 m 1, \ldots, x_1 m x_2]$.

Let $\Phi[x_1, \ldots, x_n]$ be a [illegible]-formula over the assertions x_{M}. We define $\Phi[\mathrm{Ch}_i[x_1, x_2]]$, for $i \in [1, 2]$, as the relation defined by Φ on $\mathrm{Ch}_i[x_1, x_2]$, i.e., by

$$\Phi[x_1, \ldots, x_n] \mid \mathrm{Ch}_i[x_1, x_2] \models \Phi[x_1, \ldots, x_n].$$

Lemma 8.11 Let Φ be a simple [illegible]. There exists a [illegible] formula

$$\Phi[x_1, \ldots, x_n],$$

such that for all $x_1, x_2 \geq 1$

$$\mathrm{Min}[\Phi[x_1, x_2]]|\Phi[x_1, x_2]| = \Phi[\mathrm{Ch}_1[x_1, x_2]],$$

and

$$\mathrm{Min}[\Phi[x_1, x_2]]|\Phi[x_1, x_2]| = \Phi[\mathrm{Ch}_2[x_1, x_2]],$$

where x_1 is the first still and x_2 is the second still of the set of $\Phi[x_1, x_2]$.

Proof. Let $\alpha(\bar{x}) := \beta(\bar{x})$ be the sole in the sentent $(\mathbb{N} \models \beta, s, \bar{x})$, and let $\alpha(\bar{x}) := \gamma(\bar{x})$ be the sole in the sentent $(\mathbb{N} \models \gamma, s, \bar{x})$. Let $y_1, y_2, y_3, z_1, z_2, z_3, z_4, \ldots, z_n$ be variables not occurring in β or γ. Then define $\delta(\bar{x})$ as the formula

$$\exists y_1 \exists y_2 \exists y_3 (\alpha(y_1) \wedge \alpha(y_2) \wedge \alpha(y_3) \wedge \neg(y_1 = y_2) \wedge \neg(y_1 = y_3) \wedge \neg(y_2 = y_3) \wedge \beta(y_1, y_2, y_3, \bar{x})).$$

Then $\mathrm{val}(y_1)$ is an $\mathbb{N}$-formula that defines the set of the two, and

$$\alpha(y_1, y_2, y_3, z_1, \ldots, z_n),$$

is the formula

$$(y_1 = \mathrm{first} \wedge \neg(y_1 = y_2 \wedge \beta'(\bar{x}))) \wedge (\neg(y_1 = \mathrm{first}) \wedge \neg(y_1 = y_2) \wedge \neg(y_1 = y_3) \wedge \neg(y_2 = y_3) \wedge \alpha(y_1, z_1, z_2, \bar{x}) \wedge (y_3 = y_2 \vee \gamma'(\bar{x}))),$$

where first is the first element of $\mathbb{N}$ and β' is obtained from β by replacing each occurrence of β by y_1 and each occurrence of $\mathrm{succ}(y_1)$ by y_2 and γ' is obtained from γ by replacing each occurrence of $\alpha(\bar{x})$ by $\alpha(y_1, z_1, z_2, \bar{x})$, and occurrence of β by y_1, and each occurrence of γ by y_2. The variables y_2 and y_3 make sure that the relation is never empty. It might happen that β defines an empty relation; then the formula would be empty. ∎

The next lemma is an immediate observation.

Lemma 4.18 Let $\beta(x_1, \ldots, x_k)$ be a $\mathbb{N}$-formula over $\sigma_\mathbb{N}$. Then any x_i and x_j greater than 1, and $1 \leq [1, \ldots, k]$,

(i) $\sigma_\mathbb{N} \models \beta(x_1, x_2) \leftrightarrow \beta(x_1, \ldots, x_k)$ and

$$x_1 \geq (x_2 + 1, \ldots, x_2 + x_3 - 1) \rightarrow (x_1, \ldots, x_{i-1}, x_1, \ldots x_k)$$

then for all $x \geq (x_2 + 1, \ldots, x_2 + x_3 - 1) \rightarrow (x_1, \ldots, x_k)$,

$$\sigma_\mathbb{N} \models \beta(x_1, x_2) \leftrightarrow \beta(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_k).$$

(ii) $\sigma_\mathbb{N} \models \beta(x_1, x_2) \leftrightarrow \beta(x_1, \ldots, x_k)$ and

$$x_1 \geq (1, \ldots, x_2 - 1) \rightarrow (x_1, \ldots, x_{i-1}, x_1, \ldots x_k)$$

then for all $x \geq (1, \ldots, x_2 - 1) \rightarrow (x_1, \ldots, x_k)$,

$$\sigma_\mathbb{N} \models \beta(x_1, x_2) \leftrightarrow \beta(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_k).$$

Proof. We follow from the fact that $\mathbb{N}$ (the sing Dusual logic) semant Dialog-gebell Dolmen elements that are enumerable. Clearly, the isomorphism of sing two elements $x, x' \geq (x_2 + 1, \ldots, x_2 + x_3 - 1)$ is an automorphism of $\sigma_\mathbb{N}(x_1, x_2)$, and the isomorphism of sing two elements $x, x' \geq (1, \ldots, x_2 - 1)$ is an automorphism of $\sigma_\mathbb{N}(x_1, x_2)$. ∎

Definition 6.10 Let $\{x_1, \ldots, x_k\}$ be a QBF-formula over the variables x_i. Let x_1 and x_2 be two integers greater than 1. Let $x_{1,2}$ be the variables $\{V, 0, 1, x_1, x_2\}$, containing all the constant symbols $V, 0, 1$ and the 2-ary relation symbols x_1 and x_2. Define $\Phi\{\Box, x_1, x_2\}$ as the $x_{1,2}$-structure with formula $\{1, \ldots, x_1 \cdot x_2 \cdot 1\}$, where $V = x_1 \cdot x_2 \cdot 1$, $0 = x_1$, $1 = x_1 \cdot x_2$, $x_1 = \{(V_1, x_1)\}$, and $x_2 = \{(V_2, x_2)\}$.

We now show that, under certain assumptions, on the structures $\Phi\{\Box, x_1, x_2\}$ among QBF-formulas are the equivalence formulas that essentially equal only about the relations x_1 or only about the relations x_2, that are about Until.

Lemma 6.11 Let $\Box$ be a QBF-formula with 1 free variable. Denote then as QBF-formulas $V_1[x]$ and $V_2[x]$ such that for all $x_1, x_2 \cdot 1$,

$$\Phi\{\Box, x_1, x_2\} \models V_1[x] \quad \text{iff} \quad x \in \{1, \ldots, x_1 - 1\},$$

and

$$\Phi\{\Box, x_1, x_2\} \models V_2[x] \quad \text{iff} \quad x \in \{x_1 \cdot 1, \ldots, x_1 \cdot x_2 - 1\}.$$

For every QBF-formula $\varphi[x_1, \ldots, x_k]$ over the variables $x_{1,2}$, there exists a formula $\Psi[x_1, \ldots, x_k]$ of QBF-formulas of the form

$$\varphi[x_1, \ldots, x_k] \equiv \Psi[x_1, \ldots, x_k] \equiv \bigwedge_{i=1}^{k} \varphi_i[x_i],$$

where φ from multiple contains V_1 over x_1, Ψ from multiple contains V_2 over x_2, and each $\varphi_i[x_i]$ is of the form $x_i = V$, $x_i = 0$, $x_i = 1$, $V_1[x_i]$ or $V_2[x_i]$. For every $x_1, x_2 \cdot 1$, and for every $x_1, \ldots, x_k \in \{1, \ldots, x_1 \cdot x_2 \cdot 1\}$ it holds that

$$\Phi\{\Box, x_1, x_2\} \models \varphi[x_1, \ldots, x_k] \quad \text{iff} \quad \Phi\{\Box, x_1, x_2\} \models \Psi[x_1, \ldots, x_k].$$

Proof. The proof uses the structure of φ. We use induction w.r.t. φ. That contains only suppose to check formulas that are equivalent.

1. If φ equals $x = x'$, where $x, x' \in \{V, 0, 1\}$ and $x \neq x'$, then Ψ is the empty disjunction.

2. If φ equals $x = x$, where $x \in \{V, 0, 1\}$, then Ψ is $\{\text{true} \in \text{true}\}$.

3. Suppose $\varphi[x]$ is $x = x$ or $x = V$, where $x \in \{V, 0, 1\}$. Then Ψ is $\{\text{true} \in \text{true} \in x = x\}$.

4. Suppose $\varphi[x_1, x_2]$ is $x_1 = x_2$. We that interchange some variables. Define Π as the set of symbols $\{V, 0, 1, V_1, V_2\}$. Let φ be a natural number. For any $\bar{\Box} = \Box_1, \ldots, \Box_\varphi \in \Pi^\varphi$, and variables $\bar{x} = x_1, \ldots, x_\varphi$, define for $j = 1, \ldots, \varphi$, the formula $\varphi_{\bar{\Box},j}[\bar{x}]$ as

$$\varphi_{\bar{\Box},j}[\bar{x}] := \begin{cases} x_j = V & \text{if } \Box_j = V, \\ x_j = 0 & \text{if } \Box_j = 0, \\ x_j = 1 & \text{if } \Box_j = 1, \\ V_1[x_j] & \text{if } \Box_j = V_1, \\ V_2[x_j] & \text{if } \Box_j = V_2. \end{cases}$$

Now define

$$[w_1, w_2] := \sum_{\bar{a} \in \mathbb{Z}^n} \left(w_1 - w_2 + w_1 - w_2 + \sum_{j=1}^{n} a_{\bar{a},j} [w_1, w_2] \right).$$

1. Suppose $g[w_1, \ldots, w_n]$ is $n_1[w_1, \ldots, w_n]$. Then define

$$\Phi := \sum_{\bar{a} \in \mathbb{Z}^n} \left(n_1[w_1, \ldots, w_n] + \text{lower} + \sum_{j=1}^{n} a_{\bar{a},j} [\bar{w}] \right).$$

2. Suppose $g[w_1, \ldots, w_n]$ is $n_1[w_1, \ldots, w_n]$. Then is symmetric in $[\Phi]$.

3. Suppose $g[w_1, \ldots, w_n]$ is $g_1[u_1, \ldots, u_r] + g_2[w_1, \ldots, w_n]$. Then,

$$[u_1, \ldots, u_r] + [w_1, \ldots, w_n] = [w_1, \ldots, w_n].$$

But

$$[w_1, \ldots, w_n] = [u_1, \ldots, u_r] - [w_1, \ldots, w_n]$$

and let

$$[w_1, \ldots, w_n] = [w_1, \ldots, w_n] - [u_1, \ldots, u_r].$$

By the inductive hypothesis, there exists a Φ_1 equivalent to g_1 of the form

$$\sum_{i=1}^{n} \left(w_i^1 + w_i^2 + \sum_{j=1}^{n_r} a_{i,j}^1 \right),$$

and a Φ_2 equivalent to g_2 of the form

$$\sum_{i=1}^{n} \left(w_i^1 + w_i^2 + \sum_{j=1}^{n_s} a_{i,j}^2 \right).$$

The formula Φ is obtained from Φ_1 and Φ_2, by replacing every Hadamard $w_i^1 + w_i^2 + \sum_{j=1}^{n_r} a_{i,j}^1$ in Φ_1 by

$$\sum_{\bar{a} \in \mathbb{Z}^n} \left(w_i^1 + w_i^2 + \sum_{j=1}^{n_r} a_{i,j}^1 + \sum_{j=1}^{n} a_{\bar{a},j} [\bar{w}] \right),$$

and by replacing every Hadamard $w_i^2 + w_i^1 + \sum_{j=1}^{n_s} a_{i,j}^2$ in Φ_2 by

$$\sum_{\bar{a} \in \mathbb{Z}^n} \left(w_i^2 + w_i^1 + \sum_{j=1}^{n_s} a_{i,j}^2 + \sum_{j=1}^{n} a_{\bar{a},j} [\bar{w}] \right).$$

2. Suppose $g(x_1, \ldots, x_n)$ is $-g'(x_1, \ldots, x_n)$. By the induction hypothesis, there is a $\mathbb{P}'$ equivalent to g' of the form

$$\bigwedge_{i=1}^{m} \left[\bigvee_{j=1}^{k} \left[x'_i \vee y'_i \vee \bigwedge_{l=1}^{r} z'_{i,l} \right] \right].$$

Then $-g'$ is equivalent to

$$\bigvee_{i=1}^{m} \left[\bigwedge_{j=1}^{k} \left[-x'_i \wedge -y'_i \wedge \bigvee_{l=1}^{r} -z'_{i,l} \right] \right].$$

We now transform this formula to an equivalent one in the right form. Replace each $-z'_{i,l}$ of the form

$$\begin{aligned}
(x) &= (x_j = 0) \text{ for } x_j = 1 \text{ or } x_j = 3 \text{ or } x_j = 5 \text{ or } x_k(x_j) \text{ or } x_k(x_j)),\\
(0) &= (x_j = 1) \text{ for } x_j = 0 \text{ or } x_j = 3 \text{ or } x_k(x_j) \text{ or } x_k(x_j)),\\
(x) &= (x_j = 3) \text{ for } x_j = 0 \text{ or } x_j = 1 \text{ or } x_k(x_j) \text{ or } x_k(x_j)),\\
(0) &= x_k(x_j) \text{ for } x_j = 0 \text{ or } x_j = 1 \text{ or } x_j = 3 \text{ or } x_k(x_j)),\\
(x) &= x_k(x_j) \text{ for } x_j = 0 \text{ or } x_j = 1 \text{ or } x_j = 3 \text{ or } x_k(x_j)).
\end{aligned}$$

Let the resulting formula be disjunctive normal form (there the literals are the formulas $-x'_i$, $-y'_i$, $x = 0$, $x = 1$, $x = 3$, $x_k(x)$ and $x_k(x)$). Call disjunct now looks like

$$-x_{i_1} \wedge \ldots \wedge -x_{i_a} \wedge -y_{i_1} \wedge \ldots \wedge -y_{i_a} \wedge z_1 \wedge \ldots \wedge z_b,$$

where each z is of the form $x = 0$, $x = 1$, $x = 3$, $x_k(x)$ or $x_k(x)$. The disjunct is dissatisfied if there are disjunct i's for the same variable (e.g., one is $x = 0$ and the other is $x_k(x)$). Otherwise, define x as the formula $-x_{i_1} \wedge \ldots \wedge -x_{i_a}$, and define y as the formula $-y_{i_1} \wedge \ldots \wedge -y_{i_a}$. Let $z_1, \ldots, z_b$ be the variables in $\{x_1, \ldots, x_n\}$ for which there is no z in the disjunct. Then replace the disjunct by

$$\bigvee_{\bar{i}_1 \ldots \bar{i}_b} \left[x \vee y \vee \bigwedge_{j=1}^{b} z_j \vee \bigwedge_{j=1}^{b} x_{\bar{i}_j}(\bar{i}_j) \right].$$

3. Suppose $g(x_1, \ldots, x_n)$ is $(\forall x_{n+1})(g'(x_1, \ldots, x_n, x_{n+1}))$. By the induction hypothesis, there is a $\mathbb{P}'$ equivalent to g' of the form

$$\bigwedge_{i=1}^{m} \left[\bigvee_{j=1}^{k} \left[x'_i \vee y'_i \vee \bigwedge_{l=1}^{r} z'_{i,l} \right] \right].$$

Then g is equivalent to

$$\bigwedge_{i=1}^{m} (\forall x_{n+1}) \left[\bigvee_{j=1}^{k} \left[x'_i \vee y'_i \vee \bigwedge_{l=1}^{r} z'_{i,l} \right] \right].$$

Ella le ilua ugalaulaal la

$$\sum_{i=1}^{m}\sum_{s=1}^{d}\left\|m'_{i,s}-\mathbb{E}'_{i,s}-\sum_{j=1}^{d}m'_{i,j}\right\| \tag{8}$$

$$=\sum_{i=1}^{m}|\mathbb{E}w_{m+1}|\left\|\sum_{s=1}^{d}|w_s\oplus w_{m+1}|-m'_i-\mathbb{E}'_i-\sum_{j=1}^{d}m'_{i,j}\right\|. \tag{9}$$

Hara $m'_{i,s}$ la bala lil $m'_{i,s}\oplus m'_{i,m+1}$, allaraala li agaala ila baraala illal la allalaall hara m'_i lig aaplaalag aall aaaraaaa ailila aaalalla w_{m+1} lig w_s.

Ella aallbaraala (8) la alaalig la ila algli hara. Ha aall lilqaaal 1 al (8),

(a) Lil $m'_{i,m+1}$ la $w_{m+1}=0$ ilaa halla m_i aa

$$|\mathbb{E}w_{m+1}|\left|\sum_{s=1}^{d}|w_s\oplus w_{m+1}|-w_{m+1}-0-m'_i\right|,$$

$\mathbb{E}_i$ aa

$$|\mathbb{E}w_{m+1}|\left|\sum_{s=1}^{d}|w_s\oplus w_{m+1}|-w_{m+1}-0-\mathbb{E}'_i\right|,$$

aall ha $j=1,\ldots,d$, halla $m_{i,j}$ aa $m'_{i,j}$.

(b) Lil $m'_{i,m+1}$ la $\mathbb{M}_k(w_{m+1})$ ilaa halla m_i aa

$$|\mathbb{E}w_{m+1}|\left|\sum_{s=1}^{d}|w_s\oplus w_{m+1}|-\mathbb{M}_k(w_{m+1})-m'_i\right|,$$

$\mathbb{E}_i$ aa

$$\begin{aligned}&|\mathbb{E}w_{m+1}|\left|\sum_{s=1}^{d}|w_s\oplus w_{m+1}|\right.\\&\qquad\left.-\mathbb{M}_k(w_{m+1})-w_{m+1}\oplus 0-w_{m+1}\oplus 1-w_{m+1}\oplus 2-\mathbb{E}'_i\right|,\end{aligned}$$

(ila aaaaalaaaa halla hara Haaaaa 5.18(iii)), aall ha $j=1,\ldots,d$, halla $m_{i,j}$ aa $m'_{i,j}$.

(c) Lil $m'_{i,m+1}=\mathbb{M}_k(w_{m+1})$ ilaa halla m_i aa

$$\begin{aligned}&|\mathbb{E}w_{m+1}|\left|\sum_{s=1}^{d}|w_s\oplus w_{m+1}|\right.\\&\qquad\left.-\mathbb{M}_k(w_{m+1})-w_{m+1}\oplus 0-w_{m+1}\oplus 1-w_{m+1}\oplus 2-m'_i\right|,\end{aligned}$$

(ila aaaaalaaaa halla hara Haaaaa 5.18(iii)), $\mathbb{E}_i$ aa

$$|\mathbb{E}w_{m+1}|\left|\sum_{s=1}^{d}w_s\oplus w_{m+1}-\mathbb{M}_k(w_{m+1})-\mathbb{E}'_i\right|,$$

aall ha $j=1,\ldots,d$, halla $m_{i,j}$ aa $m'_{i,j}$. ∎

In the next lemma we prove that the BB-formula are only equal about x_1 (respectively x_2) then the formula is general enough. Indeed $\varphi(\mathsf{BB}_1(x_1, x_2))$ and $\varphi(\mathsf{BB}_1(x_1, x_2))$ (respectively $\varphi(\mathsf{BB}_1(x_1, x_2))$ and $\varphi(\mathsf{BB}_1(x_1, x_2))$), where $x_1 \neq x_2$.

Lemma 4.10 Let $\varphi(x_1, \ldots, x_n)$ be a BBB-formula with the variables x_0. Suppose φ contains only as bound variables.

(i) Let ψ be an BB-subformula with the variables $x_{1,2}$ that contains only as' bound variables and that does not contain the solution equal x_1. Let $x_1 \equiv x \equiv x'$. Then for all $x_1, x_1' \equiv x \equiv x'$,

$$\mathfrak{A}[\mathbb{N}, x_1, x_1] \models \psi \quad \Longleftrightarrow \quad \mathfrak{A}[\mathbb{N}, x_1, x_1'] \models \psi.$$

(ii) Let ψ be an BB-subformula with the variables $x_{1,2}$ that contains only as' bound variables and that does not contain the solution equal x_1. Let $x_1 \equiv x \equiv x'$. Then for all $x_1, x_1' \equiv x \equiv x'$,

$$\mathfrak{A}[\mathbb{N}, x_1, x_1] \models \psi \quad \Longleftrightarrow \quad \mathfrak{A}[\mathbb{N}, x_1', x_1] \models \psi.$$

Proof. We only prove (i), (ii) is symmetric. We use induction w.r.t.g. that φ and ψ have no variables in common. Let $\mathfrak{A}[\mathbb{N}, x_1, x_1]^{\mathsf{ss}}$ be the structure $\mathfrak{A}[\mathbb{N}, x_1, x_1]$ restricted to N, S, B and x_1. Since ψ does not equal about x_1, it suffices to prove that for all $x_1, x_1, x_1' \equiv x \equiv x'$:

$$\mathfrak{A}[\mathbb{N}, x_1, x_1]^{\mathsf{ss}} \models \psi \quad \Longleftrightarrow \quad \mathfrak{A}[\mathbb{N}, x_1, x_1']^{\mathsf{ss}} \models \psi.$$

Suppose there exist $x_1, x_1' \equiv x \equiv x'$ such that

$$\mathfrak{A}[\mathbb{N}, x_1, x_1]^{\mathsf{ss}} \models \psi \quad \text{and} \quad \mathfrak{A}[\mathbb{N}, x_1, x_1']^{\mathsf{ss}} \not\models \psi.$$

Let ψ' be the formula obtained from ψ by replacing each atomic subformula $x_1[x_1, \ldots, x_n]$ by $\mathsf{B}[x_1, \ldots, x_n]$. Hence, $\mathfrak{A}_{\mathsf{B}}[x_1, x_1] \models \psi'$ and $\mathfrak{A}_{\mathsf{B}}[x_1, x_1'] \not\models \psi'$. Thus, there exists a variance with $x \equiv x'$ satisfies that distinguishes between $\mathfrak{A}_{\mathsf{B}}[x_1, x_1]$ and $\mathfrak{A}_{\mathsf{B}}[x_1, x_1']$. However, for $x_1, x_1, x_1' \equiv x \equiv x'$, using pebble games [18, 9] it is easy to show that $\mathfrak{A}_{\mathsf{B}}[x_1, x_1]$ and $\mathfrak{A}_{\mathsf{B}}[x_1, x_1']$ are indistinguishable in FOC with $x \equiv x'$ variables. This leads to the desired contradiction. ∎

By putting all the pieces together we obtain the following theorem:

Theorem 4.12 The prenex equal_subtree is not expressible by a quantifier BB.

Proof. Consider a contradiction suppose equal_subtree is expressible by a quantifier BBB. Suppose M has cardinality $n_1, \ldots, n_k$ for the prenexes equal M, and cardinality model, $\mathsf{I}_1, \ldots, \mathsf{I}_k$ for the dual equal M. W.l.o.g., we can assume that n_1 is a not-minimal cardinality that satisfies for each cardinality for predicates M and I define $n_1[\mathsf{M}] := [\mathsf{M}, \mathsf{I}]$, and for predicates M and N define $n_1[\mathsf{M}] := [\mathsf{N}] \cup n_1[\mathsf{M}]$.

[redacted] [redacted] [redacted] [

and $\mathcal{M}_k(\alpha)$ as

$$\mathcal{M}_k(\alpha) := \{\bar{\alpha}\}\{\bar{\alpha}\}\{\alpha_i(\bar{\alpha}, \alpha, \bar{\alpha}) \mid \alpha \neq 0 \text{ or } \alpha \neq 1\}.$$

Then for all $\alpha_1, \alpha_2 \in 1$,

$$\Delta\{\Gamma, \alpha_1, \alpha_2\} \models \mathcal{M}_k(\alpha) \quad \text{iff} \quad \alpha \in \{1, \ldots, \alpha_1 - 1\},$$

and

$$\Delta\{\Gamma, \alpha_1, \alpha_2\} \models \mathcal{M}_k(\alpha) \quad \text{iff} \quad \alpha \in \{\alpha_2 + 1, \ldots, \alpha_1 + \alpha_2 - 1\}.$$

Hence, by Lemma 5.15, Γ' is equivalent to a conjunction of the form $\bigwedge_{i=1}^{n} \alpha_i \in \Gamma_i$, where the α_i's are constants that do not contain α_1, and the Γ_i's are conjunctions that do not contain α_1 (since there are no free variables, there are no α's). Q.E.D., we see that if Γ and $\bigwedge_{i=1}^{n} \alpha_i \in \Gamma_i$ have no variables in common. But we do the number of variables in Γ, and let α' be the number of variables in $\bigwedge_{i=1}^{n} \alpha_i \in \Gamma_i$. By a simple counting argument, there have to exist $\alpha'_1 \in \alpha_1 \Gamma$ so $\Gamma \in \alpha'$ such that for all $i = 1, \ldots, n$

$$\Delta\{\Gamma, \alpha_1, \alpha \in \alpha'\} \models \alpha_i \quad \text{iff} \quad \Delta\{\Gamma, \alpha'_1, \alpha \in \alpha'\} \models \alpha_i.$$

Hence, by applying Lemma 5.10(i) twice, for $i = 1, \ldots, n$,

$$\Delta\{\Gamma, \alpha_1, \alpha_2\} \models \alpha_i \quad \text{iff} \quad \Delta\{\Gamma, \alpha'_1, \alpha_2\} \models \alpha_i.$$

From Lemma 5.10(ii), it follows that for $i = 1, \ldots, n$,

$$\Delta\{\Gamma, \alpha_1, \alpha_2\} \models \Gamma_i \quad \text{iff} \quad \Delta\{\Gamma, \alpha'_1, \alpha_2\} \models \Gamma_i.$$

Hence,

$$\Delta\{\Gamma, \alpha_1, \alpha_2\} \models \bigwedge_{i=1}^{n} \alpha_i \in \Gamma_i \quad \text{iff} \quad \Delta\{\Gamma, \alpha'_1, \alpha_2\} \models \bigwedge_{i=1}^{n} \alpha_i \in \Gamma_i.$$

But then by (5), we have that

$$\mathcal{M}'(\Delta\{\alpha_1, \alpha_2\}\{\mathrm{cond}(\alpha)\}) \text{ is true} \quad \text{iff} \quad \mathcal{M}'(\Delta\{\alpha'_1, \alpha_2\}\{\mathrm{cond}(\alpha)\}) \text{ is true},$$

and $\alpha_1 \neq \alpha'_1$. Hence, $\mathcal{M}'$ does not support $\mathrm{good\_cond}$. ∎

5.5 Finite model theorem

We have alternatively shown, in the many sorted Datalog in MDD (Theorem 5.4), and MDDs in the general (multisorted) Datalog in DDD-DIS (Theorem 5.5). It remains to compare these two formalisms with each other. We will show that equivalently MDDs are actually strictly more powerful than DDD.

DDD are in fact in the standard way to define general of any order.

Definition 5.17 Let $g(\alpha_1, \ldots, \alpha_n)$ be an DDD-formula. Then g defines the k-ary query $\mathcal{Q}$ defined by

$$\mathcal{Q}(\mathcal{I}) := \{(\alpha_1, \ldots, \alpha_n) \mid \mathcal{I} \models g(\alpha_1, \ldots, \alpha_n)\},$$

for every form $\mathcal{I}$.

Theorem 6.10 Every 0-ary gate definable in [illegible] is expressible by a [illegible] using only equational utilization.

Proof. Consider an [illegible]-formula $\varphi(x_1, \ldots, x_n)$. For any tree t and values $x_1, \ldots, x_n$ of t, we use also the tuple $(t, x_1, \ldots, x_n)$ as a labeling of t with elements of $\{0, 1\}^n$ by labeling a node v with $x_1 \ldots x_n$ such that for $i = 1, \ldots, n$,

$$x_i = \begin{cases} 1 & \text{if } v = x_i, \\ 0 & \text{otherwise.} \end{cases}$$

An $(t, x_1, \ldots, x_n)$ is a Σ^n_Σ-tree as defined in Section 5.1. It is easy to write an [illegible](Σ^n_Σ)-sentence Φ such that for every finite tree t and values $x_1, \ldots, x_n$ of t:

$$(t, x_1, \ldots, x_n) \models \Phi \quad \text{iff} \quad t \models \varphi(x_1, \ldots, x_n).$$

Indeed, for $i = 1, \ldots, n$, define Φ_i as the set of pairwise equivalent for which the i-th component is 1, i.e.,

$$\Phi_i := \{ \bar{v} \mid \bar{v} \in \Phi, \bar{v} \in \{0, 1\}^n \text{ and } v_i = 1 \}.$$

Then Φ is defined as

$$\{\Phi_1\} \ldots \{\Phi_n\}$$

$$\left[\left[\left[\bigcup_{\bar{v} \in \Phi_1} \Phi_{\bar{v}}(x_1)\right] \cap \ldots \cap \left[\bigcup_{\bar{v} \in \Phi_n} \Phi_{\bar{v}}(x_n)\right]\right]\right] = \varphi'(x_1, \ldots, x_n)\Big],$$

where φ' is the formula obtained by replacing each atomic formula $\Phi_{\bar{v}}(x)$ in φ by $\bigcup_{\bar{v} \in \{0,1\}^n} \Phi_{\bar{v}}(x)$.

By Theorem 5.5 there exists a tree automaton $\mathcal{A}$ (defined over Σ^n_Σ) that accepts only those $(t, x_1, \ldots, x_n)$ such that $(t, x_1, \ldots, x_n) \models \Phi$, where t is a finite tree. Hence, the theorem is proved if we can construct a [illegible] [illegible], we call finite tree t, elements [illegible] in parallel on all possible labelings of t, returning those labelings $(x_1, \ldots, x_n)$ such that $(t, x_1, \ldots, x_n)$ is accepted by [illegible]. Clearly, we use a 0-ary relation-valued equational utilization [illegible] that stands for [illegible]. The semantic value we call that for each node v, $(x_1, \ldots, x_n) \in (x)$ [illegible] accepts state [illegible] as node v in the execution on $(t, x_1, \ldots, x_n)$. The utilization eval at the root is then defined as $\mathcal{A}_{\mathrm{run}}$, where [illegible] is the set of final states of [illegible].

The formula $\mathrm{eval}_{\mathcal{A}}(x_1, \ldots, x_n, y)$ defines the labelings $(x_1, \ldots, x_n)$ of the tree t such that node y is labeled with $\bar{v}$:

$$\mathrm{eval}_{\mathcal{A}}(x_1, \ldots, x_n, y) := \bigcup_{x_1 = 1} x_1 - y \cap \bigcup_{x_1 = 1} x_1 \cup y.$$

Let $w = \ldots$ [illegible] $\ldots$ be a predecessor of [illegible]. Define $\mathrm{S}(w) := \{ v \in \{1, \ldots, n\} \mid$ [illegible] is a terminal$\}$ and $\mathrm{P}(w) := \{1, \ldots, n\} - \mathrm{S}(w)$. For each v define the

[illegible]

$$
\begin{aligned}
[illegible] &:= \\
&\Big| [illegible]_{\bar{a}}[\bar{c}, 0] = \sum_{[illegible]} [illegible]_{\bar{a}}[\bar{c}, 1] = \mathbb{1}[|\bar{c}|] = \sum_{[illegible]} m_1[\bar{c}, 1] \Big| \\
&\quad \bar{c}_1, \ldots, \bar{c}_n \in [0, 1]^n, \\
&\quad 1, \ldots, n \in \quad , \\
&\quad \forall i \quad \bar{c}_i = \quad \bar{c}_1 \ldots \quad \bar{c}_n, \quad 1, \ldots, n] = \quad],
\end{aligned}
$$

where $m_1[\bar{c}, n]$ is the formula $[illegible]_{\bar{a}}[\bar{c}, n] \ [illegible] \ \bar{c}_1] = \quad$, and is false otherwise.

The correctness of this construction now follows from the following lemma, which is easily proven by induction on the length of n.

Lemma 4.1 Let Φ be a tree. We call node v of Φ, $[m_1, \ldots, m_k] \in [n] \times [illegible]$ [illegible] on node v in the execution on the input tree $[\Phi, m_1, \ldots, m_k]$.

This completes the proof. ∎

The proof of the previous theorem, like the proof of Theorem 3.3, involves the simulation in parallel of a tree automaton on different inputs of a tree. However here, the simulation is much more straightforward since we are simply propagating the simulation, along relation-valued situations. This was not possible in the case of [illegible] [Theorem 3.3] which dealt only Boolean-valued situations; the simulation there was much more intricate. In particular, while Theorem 3.13 states that equivalent situations are enough for a [illegible] to improve all of [illegible], this is not the case for [illegible]. Instead we replicated it the set of Lemma 3.3, [illegible] only equivalent situations are needed there [illegible].

We finally show that equivalent [illegible] are strictly more powerful than [illegible]. However, this only holds when the assumption that the subjoining process are performed on infinite monoids of finite trees.

Definition 4.20 A process is extended if the monoid of its finite trees is infinite.

Clearly, if the process is not extended, i.e., the monoid of finite trees is finite, then [illegible] and [illegible] are equally powerful and collapse to first-order logic. We need show:

Theorem 4.21 There are extended processes, equivalent [illegible] are expressive Boolean queries not definable in [illegible].

Proof. Consider an extended process Π. Since which is expressive of polynomial $\bar{\mathbf{w}} = \mathbf{w}_1, \ldots, \mathbf{w}_m$, is expressive of monoid $\bar{\mathbf{D}} = \mathbf{D}_1, \ldots, \mathbf{D}_m$, and is non-trivial , and that [i] is the left-hand side of $\mathbf{w}_1$ and assume is position $\mathbf{D}_m$ of the right-hand side of $\mathbf{w}_m$; [ii] for $i = 1, \ldots, m - 1$ the symbol in position $\mathbf{D}_i$ of the right-hand side of $\mathbf{w}_i$ equals the non-terminal on the left-hand side of $\mathbf{w}_{i+1}$; [iii] the left-hand sides of $\mathbf{w}_1, \ldots, \mathbf{w}_m$ are mutually distinct; and [iv] is reachable from the start symbol.

Consider a derivation tree $\mathcal{T}$ of Φ. We say that a node v_i is an ancestor of $(\bar{w}, \bar{\Omega})$ in $\mathcal{T}$ if there exists a sequence of nodes $v_1, \ldots, v_m, v_{m+1}$ such that for $i = 1, \ldots, m$, v_i is derived with g_i and v_{i+1} is the Ω_i-th child of v_i. We say that v_{m+1} is the left of the ancestor v_1. Note that v_{m+1} is labeled with . We call a sequence of nodes $v_1, \ldots, v_s$ a chain of ancestors of $(\bar{w}, \bar{\Omega})$ if for each $i = 1, \ldots, s-1$, v_i is an ancestor, v_{i+1} is the left of the ancestor v_i, and v_s is not derived with g_1. The length of the chain of ancestors $v_1, \ldots, v_s$ is s.

Let $\mathcal{T}$ be the Earley parse tree derived as follows: an Earley derivation tree $\mathcal{T}$, $(\bar{w}, \bar{\Omega})$ is here if there is a chain of ancestors starting at the first -labeled node in the preorder traversal of the tree and the length is a power of two. Note that $\mathcal{T}$ is here as any tree where there is an -labeled node.

The proof is not difficult in EEE. We now show that is not impossible in EEE. Let g be an EEE-grammar. By Theorem 1.1 there exists a tree substitution $\mathcal{T}' = (\ , \Omega', \mathcal{T}')$ satisfying properly the tree exhibiting g. Consider a tree $\mathcal{T}$ such that the length, which we denote by s, of the chain of ancestors of $(\bar{w}, \bar{\Omega})$ starting in the first -labeled node in the preorder traversal of the tree is a power of two. Note that $\mathcal{T}$ is here as any tree where there is an -labeled node.

The proof is not difficult in EEE. We now show that is not impossible in EEE. Let g be an EEE-grammar. By Theorem 1.1 there exists a tree substitution $\mathcal{T}' = (\ , \Omega', \mathcal{T}')$ satisfying properly the tree exhibiting g. Consider a tree $\mathcal{T}$ such that s is a power of two and s is bigger than $|\ | = 1$. Where then is the tree where s and s' of $\mathcal{T}$, such that s' is a descendant of s, that are assignments of $(\bar{w}, \bar{\Omega})$ in the chain, and $\bar{\Omega}[s][\ ] = \bar{\Omega}[s'][\ ]$, where $\Omega[s]$ denotes the children of s with root s. Let s be the v_s-th assignment and s' be the $v_{s'}$-th assignment in the chain. Let s' be the tree attached from s by replacing the children $\Omega[s']$ by the children $\Omega[s]$. Where the chain of assignments of $(\bar{w}, \bar{\Omega})$ starting in the first -labeled node in the preorder traversal of s' has length $s + v_s - v_{s'}$. This is not a power of two because $s + v_s - v_{s'}$ is $v_s - v_{s'}$ is. However, $\bar{\Omega}[s] = \bar{\Omega}[s']$, and thus $s' \models g$ if and only if $s \models g$. Hence, g does not define .

The proof is complete if is a minimal EEE. The EEE can the following equivalent situations for all non-terminals:

1. In a Earley situation: $[s]$ is here if there is a node labeled among the descendants of s $[$note that is a non-terminal$]$;

2. chain is a Earley situation:

   $[a]$ chain$[s]$ is false if there is an -labeled node among the descendants of s, or if there is a chain of assignments starting at the first -labeled node in the preorder traversal of the children with root s;

   $[b]$ chain$[s]$ is true if the length of the chain of assignments of $(\bar{w}, \bar{\Omega})$ starting at the first -labeled node in the preorder traversal of the children with root s is a power of two. Note that if this node is not labeled by g_1, then the length of the chain of assignments starting at that node is 1, which is a power of two.

3. $\mathcal{M}$ is a pre-chain situation: $\mathcal{M}[s]$ contains s and all descendants of s.

4. $\mathcal{R}$ is a Kleene situation: $\mathcal{R}[s]$ is a total order on $\mathcal{M}[s]$.

1. [illegible] is a Boolean situation: [illegible] is true iff [illegible] is the Boolean value w_1, for some $1 \in \{1, \ldots, m\}$, and there exist value vectors $v_{111}, \ldots, v_{m11}$, such that a state of assignments starts at v_{m11}, such that $j = 1$ iff $1, \ldots, m$, v_j is Boolean value w_j, and v_{j11} is the D_j-ID string of v_j.

2. In-w_1 is a Boolean situation: In-$w_1[v]$ is true iff v is Boolean value w_1.

3. n is a not-valued situation: $n[v]$ is a subset of $V[v]$.

   [a] If [illegible] is false, then $n[v]$ is empty;

   [b] If [illegible] is true and v is not Boolean for w_1, then the value for $n[v]$ varies, w.r.t. $V[v]$, by giving the length of the state of assignments of $[\bar{w}, \bar{D}]$ starting at v_{111}, where v_{111} is an Boolean for 1; if $n[v]$ contains the value $v_1, \ldots, v_n$, and there exists corresponding to the 1_1-ID, $\ldots$, 1_n-ID pairs in the ordering $V[v]$, then $n[v]$ varies the number $\sum_{i=1}^{n} 2^{l_i - 1}$.

   [c] If [illegible] is true and v is Boolean for w_1, then the value for $n[v]$ varies, w.r.t. $V[v]$, by giving the length of the state of assignments of $[\bar{w}, \bar{D}]$ starting at v.

The [illegible] is now defined as follows:

1. Consider the predicates $p = \ldots$ and $\ldots$ and $\ldots$ and $\bar{w}$. Define $S[p] = \{1 \in \{1, \ldots, m\} \mid \ldots \text{ is true}\}$. The order $1 \in S[p]$ is a dictionary for $1 \in \{1, \ldots, m\} - S[p]$. Define

$$[\bar{w}] := \begin{cases} \text{true} & \text{if } \ldots = \ldots \\ \text{[illegible]} & \text{otherwise;} \end{cases}$$

and

$$\text{state}[\bar{w}] := \begin{cases} \text{true} & \text{if } \ldots = \ldots \\ v & \text{otherwise,} \end{cases}$$

where v is an [illegible]-assignment where [illegible] value equals that of state$[\bar{w}]$, where 1 is the smallest such that $[\bar{w}]$ is true; if [illegible] 1 does not exists, v is false. For $1 = 1, \ldots, m$, let $S[1]$ be $V[1]$ if $1 \in S[p]$, and $\{[1]\}$ where $1 \notin S[p]$. Define Define

$$V[\bar{w}] := \bigcup_{1=1}^{m} S[1] \cup \{[\bar{w}]\}$$

$$v[\bar{w}] := \bigcup_{1 \in S[p]} v[1] \cup \{V[\bar{w}]\} \cup \{[\bar{w}]\} \cup \bigcup_{1 \notin S[p]} \{[1, 1]\}$$
$$\cup \bigcup \{[v[1] \cup v[\bar{w}] \mid 1, 1 \in \{1, \ldots, m\} \cap 1 \cap 1\}$$

$$\text{[illegible]}[\bar{w}] := \text{false;}$$
$$\text{In-}w_1[\bar{w}] := \text{false;}$$

and

$$n(x) := 1.$$

If $x = 0$ then define

$$\mathrm{mult}(x) := \mathrm{ord}(x) \text{ mod } \mathrm{char}(x).$$

2. For $y_1 = x \text{ mod } p, \ldots, x_r$, define

$$f(x) := \mathrm{lcm}_i$$

and

$$\mathrm{char}(x) := \mathrm{ord}(x) \text{ if } (\mathrm{ord}(x) \text{ is } \mathrm{lcm} \text{ of } \mathrm{ord}(x_i) \text{ mod } x = x_i).$$

The attribute char has because lcm of $\mathrm{ord}(x)$ is lcm and of $n(x)$ ambiguous mostly one element, i.e., $n(x)$ measures a number which is a power of lcm. Define,

$$m(x) := \prod_{i=1}^{x_r} m(i) \text{ if } f(x)_i$$

$$\begin{aligned} m(x) := & \prod_{f(x)_r} m(i) \text{ if } \prod_{i,\ldots,x_r-f(x_r)} f(x_i) \text{ is } m(i) \\ & \text{ if } f(x) \text{ is } f(x)_i \text{ if } \prod_{f(x)_r} f(i, i) \\ & \text{ if } \prod (f(i) \text{ is } m(i') \mid \\ & \quad i, i' \text{ in } (1, \ldots, x_r), \ i \text{ in } i', \ i \neq x_r \text{ in } i' \neq x_r). \end{aligned}$$

The reason for this definition is that, in order to correctly represent the number in $n(x_r)$, we have to make sure that all elements not in $m(x_r)$ come after the elements in $m(x_r)$ in the ordering $m(x)$. Finally define

$$\mathrm{ord}(x) := \mathrm{ord}(x_r)$$
$$\text{in-}y_1(x) := \mathrm{lcm}$$

and

$$n(x) := \mathrm{ord}(x) \text{ if } g(x, n(x_r), m(x)),$$

where $g(x, y, z)$ is the formula

$$\begin{aligned} (\forall x')(f(x') \text{ if } -m(\mathrm{lcm}(x')) \text{ if } (\forall x'')(x'' \text{ if } x' \text{ or } m(x'')) \\ \text{ if } (x \text{ in } \mathrm{lcm}(x') \text{ if } (x' \text{ in } x \text{ if } m(x)))) \\ \text{ if } (\forall x')(\mathrm{final}(x') \text{ if } -m(x') \text{ if } (x \text{ in } x' \text{ if } m(x))), \end{aligned}$$

will have the successor function, and that is the first element in the ordering m. The formula suppresses the number in $n(x_r)$ with 1.

[illegible]. [illegible] $j = 1, \ldots, m - 1$, [illegible] [illegible] $m_j = \ \times \ \ldots \ m_j$,

$$[illegible] := \sum_{[illegible]} [illegible]$$

[illegible]

$$[illegible] := [illegible],$$

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible], [illegible]

$$[illegible] := \sum_{[illegible]}^{m_j} [illegible] \ [illegible]$$

$$[illegible] := \sum_{[illegible]} [illegible] \ \sum_{[illegible]} [illegible] \ [illegible]$$
$$[illegible] \ [illegible] \ \sum_{[illegible]} [illegible]$$
$$[illegible] \ \sum_{[illegible]} [illegible] \ |$$
$$[illegible], [illegible] \in [illegible], \ [illegible], \ [illegible] \neq [illegible] \ [illegible] \neq [illegible]|$$

$$[illegible] := [illegible]|$$
$$[illegible] := [illegible]|$$

[illegible]

$$[illegible] := [illegible] \ [illegible] \ [illegible].$$

[illegible]. [illegible] $m_m = \ \times \ \ldots \ m_m$, [illegible]

$$[illegible] := [illegible]|$$

[illegible]

$$[illegible] := [illegible].$$

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible],

$$[illegible] := \sum_{[illegible]}^{m_m} [illegible] \ [illegible]$$

$$[illegible] := \sum_{[illegible]} [illegible] \ \sum_{[illegible]} [illegible] \ [illegible]$$
$$[illegible] \ [illegible] \ \sum_{[illegible]} [illegible]$$
$$[illegible] \ \sum_{[illegible]} [illegible] \ |$$
$$[illegible], [illegible] \in [illegible], \ [illegible], \ [illegible] \neq [illegible] \ [illegible] \neq [illegible]|$$

$$\mathrm{succ}(x) := \mathrm{ite}_{\mathrm{Bool}}(x = \mathrm{succ}(0_{\mathrm{Bool}}),$$
$$\mathrm{ite}_{\mathrm{Bool}}(x) := \mathrm{false},$$

and define

$$n(x) := x.$$

Then, $\mathcal{A}$ is an $\mathbb{N}$-structure that represents the following: $n(x)$ contains the singleton consisting of the first element in $\mathrm{dom}(x)$ if $\mathrm{ite}_{\mathrm{Bool}}(0_{\mathrm{Bool}})$ is false; $n(x)$ equals $n(0_{\mathrm{Bool}})$ if $\mathrm{succ}(0_{\mathrm{Bool}})$ is true; and $n(x) = \emptyset$ otherwise. ■

We summarize, we note:

Corollary 4.35 *$\mathbb{NHA}$ can express some among greater than $\mathbb{NHA}$.*

5 Relational attitude in grammar

Relational attitude grammar[4] can be generalized to classical attitude grammar introduced by Barwise and Cooperrider [3]. In relational attitude grammar, the semantic value an larger quality functions, compelling attitudes to have of other attitudes, that either relations among attitudes. When there is no longer a distinction between equivalent and individual attitudes, and the values of the attitudes are no longer uniquely determined by every item. We consider relational attitude grammar in the context of $\mathbb{NHA}$ and $\mathbb{NHA}$, and discuss how they are expressive greater. We show that for $\mathbb{NHA}$ this does not increase the expressive power, while in the case of $\mathbb{NHA}$ the complexity classes $\mathbb{NP}$, $\mathbb{NEXP}$ and $\mathbb{NE}$ coincide are optimal.

5.1 Relational $\mathbb{NHA}$s

An attitude grammar consisting in one just a tuple $(\mathbb{M}, \mathbb{Att})$, where $\mathbb{M}$ is a finite set of attitudes and $\mathbb{Att}$ is a function from $\mathbb{M}$ to the powerset of $\mathbb{M}$. A relational $\mathbb{NHA}$ $\mathbb{N}$ assigns to each predicate $p = \quad_1 \ldots \quad_k$ a propositional formula φ_p over the set of propositional symbols

$$\{n(x) \mid x \in \{0, \ldots, k\}, n \in \mathbb{Att}(\quad_i)\}.$$

$\mathbb{N}$ selection of a Boolean from $\mathbb{S}$ is a mapping that assigns a truth value to each $n(x)$, where $n \in \mathbb{M}$, x is a node of $\mathbb{S}$ and n is an attitude of the label of x. Let x_1 be a node of $\mathbb{S}$ with children $x_1, \ldots, x_k$ labeled by predicate p. Let φ_p be the formula associated to p. Then define $W(\mathbb{N}, \mathbb{S}, x_1)$ as the formula obtained from φ_p by replacing each propositional symbol of the form $i(n)$ by the new propositional symbol $i(x_n)$. We continue total valuation v of $\mathbb{S}$ is said to satisfy $\mathbb{N}$ if the root internal node x $W(\mathbb{N}, \mathbb{S}, x)$ is true under v.

[4] Relational attitude grammar are not to be confused with the relation- circuit attitude grammar ($\mathbb{RAHA}$) of the previous section. In fact, in the general section, we will indeed consider relational values of $\mathbb{RAHA}$.

```
X := X      ← w_0(x)
X := XX     (w_0(x)) := 1_w(x) X w_0(x))
            X (x = y) := ← x = y))
            X (x < y) := x = y) X w_0(x))
X := X      ← x = y) X ← x < y)
X := v      1_v(x)
X := p      ← 1_v(x)
```

Figure 1: Example of a relational BNF.

A relational BNF can express many grammar in various ways. But X is a many group sell but X is a relational BNF. Malgents among the attributes of X are attributes small.

(I) X is expressed relationally by X if X has many :

$$X(x) \leftrightarrow (x \mid \text{there exists a relation } v \text{ of } x \text{ that satisfies } X, \text{ and } v(x, \ldots, x) \text{ is true});$$

(II) X is expressed universally by X if X has many :

$$X(x) \leftrightarrow (x \mid \text{for every relation } v \text{ of } x \text{ that satisfies } X, v(x, \ldots, x) \text{ is true});$$

(III) X is expressed implicitly by X if X has many : there exists exactly one relation v of x that satisfies X, and v $X(x)$ if $v(x, \ldots, x)$ is true.

We denote the class of many grammar relationally (respectively, universally and implicitly) expressible by relational BNFs, by R-BNF (respectively, U-BNF and IBNF).

Example 2.1 In Figure 1 an example of a relational BNF is displayed. It expresses relationally, universally and implicitly the same group expressed by the BNF in Example 1.1. ■

The following theorem says that going from BNFs to relational BNFs does not increase the expressive power.

Theorem 2.1 BNF ⊆ R-BNF ⊆ U-BNF ⊆ BNF.

Proof. Clearly, by Theorem 1.11, BNF ⊆ R-BNF, BNF ⊆ U-BNF and BNF ⊆ IBNF. By using Theorem 1.1, we then only have to prove that many group in R-BNF, U-BNF and IBNF is definable in BNF.

1. Let X be a group that is relationally expressed by the relational BNF X. Then X is defined by the BNF-formula $X(x) :=$

$$\left(\exists x_1 \ldots \exists x_n \left(\bigwedge_{v = X, v \in X_1 \ldots X_n} \left(\exists x_1 \ldots \exists x_n (v(x_1, \ldots, x_n) \leftrightarrow \bar{x}_1) \right) \right) \right.$$

$$\left. \wedge \exists x_1 \ldots \exists x_n \, v(x) \right)$$

where Φ is the set of attributes of $\mathcal{R}$, $p(v_1, \ldots, v_k)$ is the $\mathcal{RE}$-formula that represents that values $v_1, \ldots, v_k$ are derived by predicates p, and $\bar{\mathcal{B}}_S$ is the formula obtained from $\mathcal{B}_S$ by replacing each occurrence of $\{\}$ by $\mathcal{B}_0[v_1]$. Intuitively, the $\mathcal{B}_i$'s define relations that satisfy Φ.

2. Let Φ be a query that is syntactically represented by the relational $\mathcal{RDB}$ $\mathcal{R}$. Then Φ is defined by the $\mathcal{RDB}$-formula $\Phi[v] :=$

$$\{\mathcal{RB}_1 \ldots \mathcal{RB}_k \mid \bigvee_{S = \mathcal{B}_1 \mathcal{B}_2 \ldots \mathcal{B}_k} \{\mathcal{B}_1\} \ldots \{\mathcal{B}_k\} (\Phi(v_1, \ldots, v_k) \wedge \bar{\mathcal{B}}_S) \wedge \mathcal{B}_{\mathrm{head}}[v]\}.$$

Here $\Phi(v_1, \ldots, v_k)$ and $\bar{\mathcal{B}}_S$ are defined as in (1).

3. Let Φ be a query that is implicitly represented by the relational $\mathcal{RDB}$ $\mathcal{R}$. Then the $\mathcal{RDB}$-formula that defines Φ is the same as in (1). ■

10.3 Relational $\mathcal{RDB}$s

Each attribute a has an associated sort s_a. A relational $\mathcal{RDB}$ $\mathcal{R}$ associates to each predicate $p = \ \ _1 \ldots \ \ _k$ an $\mathcal{RE}$-relation $\mathcal{B}_p$ over the corresponding

$$\bigcup_{i=1}^{k} \{v[i] \mid v \in \mathcal{B}_{\mathrm{sort}(\ \ _i)}\} \subseteq \{0, 1, \ldots, n\},$$

where for each $i = 1, \ldots, k$, $\ \ _i$ is a constant symbol and $v[i]$ is a relation symbol of sort s_a. A relation of a Datalog$^{\neg}$ has 1 is a mapping that assigns to each $v[i]$ an s_a-sorg relation over the value of 1, where $v \in \mathcal{B}$, a is a value of 1 and v is an attribute of the label of a. Let s_p be a value of 1 with attributes $s_1, \ldots, s_k$ defined by predicate p. Let $\mathcal{B}_p$ be the $\mathcal{RE}$-relation associated to p. Then define $\mathcal{W}(\mathcal{R}, 1, s_i)$ as the $\mathcal{RE}$-relation obtained from $\mathcal{B}_p$ by replacing each occurrence of the relation symbol $\{\}$ by the relation symbol $\{v[i]\}$ by the relation symbol $\{s_i\}$ and by replacing each constant symbol $\ \ _i$ by the value s_i. A relation r of 1 is call in relation $\mathcal{R}$ if $\mathcal{W}(\mathcal{R}, 1, s)$ evaluates to true when each relation symbol $\{s\}$ in $\mathcal{W}(\mathcal{R}, 1, s)$ is interpreted by r.

A relational $\mathcal{RDB}$ can express 0-sorg queries in various ways. Let $\mathcal{R}$ be a 0-query and let $\mathcal{R}$ be a relational $\mathcal{RDB}$. Analogous among the attributes of $\mathcal{R}$ a 0-sorg attribute exists. Let r denote the set of 1.

(1) $\mathcal{R}$ is expressed relationally by $\mathcal{R}$ if for every 1

$$\mathcal{R}[r] = \{(s_1, \ldots, s_k) \mid \text{there exists a valuation } v \text{ of } 1 \text{ that satisfies } \mathcal{R}, \text{ and } (s_1, \ldots, s_k) \in v[\mathrm{head}(r)]\}.$$

(11) $\mathcal{R}$ is expressed extensionally by $\mathcal{R}$ if for every 1

$$\mathcal{R}[r] = \{(s_1, \ldots, s_k) \mid \text{for every valuation } v \text{ that satisfies } \mathcal{R} \ (s_1, \ldots, s_k) \in v[\mathrm{head}(r)]\}$$

$$
\begin{aligned}
\mathbb{M} \models \mathbb{N} \quad & \{\mathbb{R}x \mid \{\mathbb{P}'_{\mathbb{R}}\}\{\mathbb{M}\}\{\mathbb{P}\}\{x\} \models \{\mathbb{M}\}\{\mathbb{P}\}\{x, \mathbb{R}\}\} \\
& \models \{\mathbb{R}_{\mathbb{R}}\{\mathbb{P}'_{\mathbb{R}}\}\{\mathbb{M}\}\{\mathbb{P}\}\{\mathbb{R}\} \models \{\mathbb{M}\}\{\mathbb{P}\}\{x, \mathbb{R}\}\} \\
& \models \mathrm{result}\{\mathbb{R}\} \\
\mathbb{M} \models \mathbb{R} \quad & \{\mathbb{R}x \mid \{\mathbb{M}\}\{\mathbb{P}\}\{x\} \models x = \mathbb{N} \land x = \mathbb{R} \land \mathbb{M}\{\mathbb{P}\}\{x\}\} \\
\mathbb{M} \models \mathbb{S} \quad & \{\mathbb{R}x \mid \{\mathbb{M}\}\{\mathbb{P}\}\{x\} \models x = \mathbb{N} \land x = \mathbb{S}\}
\end{aligned}
$$

Figure 3: Example of a relational $\mathbb{RHL}$.

[iii] $\mathbb{M}$ is expressed implicitly by $\mathbb{M}$ $\mathbb{RHL}$ but using $\mathbb{P}$ there relate exactly one valuation v of $\mathbb{P}$ that satisfies $\mathbb{M}$, call $\{m_1, \ldots, m_n\} \in \mathbb{M}\{\mathbb{P}\}$ $\mathbb{RHL}$ $\{m_1, \ldots, m_n\} \in v\{\mathrm{result}\{\mathbb{P}\}\}$.

We denote the class of among general relationally [nondeterministic, relationally and implicitly] expressible by relational $\mathbb{RHL}$s by $\mathbb{R}$-$\mathbb{RHL}$ [nondeterministic, $\mathbb{R}$-$\mathbb{RHL}$ and $\mathbb{RHL}$].

Example 3.2 In Figure 3 an example of a relational $\mathbb{RHL}$ $\mathbb{M}$ is depicted. The $\mathbb{RHL}$ expresses relationally the Boolean query which is true for a tree if the number of nodes in its left subtree equals the number of nodes in its right subtree. Put it for a tree, a the root of $\mathbb{T}$, x_1 the left child and let x_2 be the right child of the root. For any valuation v of $\mathbb{P}$ that satisfies $\mathbb{M}$, $v\{\mathbb{M}\{x_1\}\}$ contains the nodes of the left subtree, and $v\{\mathbb{M}\{x_2\}\}$ contains the nodes of the right subtree. The postcondition consisting in the root can only be true when v if $v\{\mathbb{M}\{x\}\}$ contains a bijection between $v\{\mathbb{M}\{x_1\}\}$ and $v\{\mathbb{M}\{x_2\}\}$. ■

Clearly, by Lemma 3.1[i], any query expressible by a $\mathbb{RHL}$ is in $\mathbb{R}$-$\mathbb{RHL}$, $\mathbb{R}$-$\mathbb{RHL}$ and $\mathbb{RHL}$.

In the following theorem, we characterize the 3 classes of relational $\mathbb{RHL}$s queried in terms of the complexity classes $\mathbb{NP}$ and $\mathbb{NE}$ [and their complements]. $\mathbb{NP}$ is well known; $\mathbb{NE}$ is the class of problems decidable by a polynomial time non-deterministic Turing machine that is nondeterministic, i.e., that has at most one accepting computation for any input [11, 12].[4] We obtain the following:

Theorem 3.4 1. $\mathbb{R}$-$\mathbb{RHL}$s equals the class of queries in $\mathbb{NP}$;

2. $\mathbb{R}$-$\mathbb{RHL}$s equals the class of queries in $\mathrm{co}\mathbb{NP}$; and

3. $\mathbb{RHL}$s equals the class of queries in $\mathbb{NP} \cap \mathrm{co}\mathbb{NP}$.

Proof.

1. The relational $\mathbb{R}$-$\mathbb{RHL}$ of $\mathbb{NP}$ is clear. For the converse, we make use of Fagin's Theorem [10, 7], which states that the queries expressible in $\mathbb{NP}$ are exactly those that are definable in relational second-order logic [$\mathbb{SOL}$]. Every $\mathbb{SOL}$-formula is of the form

$$\{\mathbb{SO}_1\} \ldots \{\mathbb{SO}_n\}\{\mathbb{M}\}\{\bar{v}, \mathbb{R}_1, \ldots, \mathbb{R}_n\}\},$$

[4] The $\mathbb{R}$-ing group $\mathbb{R}$ Gulsage is a complexity class $\mathbb{C}$ if the Boolean problem $\{\{x, m_1, \ldots, m_n\} \mid \{m_1, \ldots, m_n\} \in \mathbb{M}\{x\}\}$ is in $\mathbb{C}$.

where the $\mathbb{R}_i$ are relative relative and $\mathbb{P}$ is an $\mathbb{RR}$-formula over the vocabulary augmented with the relation symbols $\{\mathbb{R}_1, \ldots, \mathbb{R}_n\}$.

Consider the $\mathbb{RRR}$-formula: $\{\mathbb{RR}_1\} \ldots \{\mathbb{RR}_n\}\{\mathbb{P}\}(\bar{\mathbb{x}}, \bar{\mathbb{y}})$. Like in the proof of Theorem 5.2, we can construct a $\mathbb{RRP}$ that computes all the relations that make up a fixpoint from scratch as a relational structure. $\mathbb{RRP}$ is the $\mathbb{RRP}$ the role for the start symbol

$$\{\mathbb{R}\bar{\mathbb{x}}\}\{\mathbb{reach}\}\{\mathbb{R}\}\{\bar{\mathbb{x}}\} \equiv \mathbb{P}'\{\bar{\mathbb{x}}\},$$

where $\mathbb{P}'$ is obtained from $\mathbb{P}$ by replacing each $\mathbb{R}_i\{\bar{\mathbb{x}}\}$ by $\mathbb{R}_i\{\mathbb{R}\}\{\bar{\mathbb{x}}\}$, and by replacing each relation of the vocabulary of the relational structure by its corresponding situation. It then follows that this relational $\mathbb{RRP}$ computes exactly the same fixpoint by $\{\mathbb{RR}_1\} \ldots \{\mathbb{RR}_n\}\{\mathbb{P}\}(\bar{\mathbb{x}}, \bar{\mathbb{y}})$.

3. We prove that $\mathbb{L}$-$\mathbb{RRP}$s are the general computation in $\mathbb{SSR}$, we make use of the complement of Gogic's Theorem: $\mathbb{SSR}$ = Universal second-order logic. The proof is then analogous to [1].

4. Finally, $\mathbb{IRRP}$ $\mathbb{P}$ $\mathbb{RR}$ $\mathbb{S}$ $\mathbb{ESR}$.

   Let $\{\mathbb{R}_1, \ldots, \mathbb{R}_n\}$ be a sequence of general. We say that the sequence

$$\{\mathbb{R}_1, \ldots, \mathbb{R}_n\}$$

   is implicitly defined in $\mathbb{RR}$ if there is an $\mathbb{RR}$-sentence $\mathbb{P}\{\mathbb{R}_1, \ldots, \mathbb{R}_n\}$ over the vocabulary of Relation from sequential with $\{\mathbb{R}_1, \ldots, \mathbb{R}_n\}$ such that for every from $\mathbb{S}$ the sequence $\{\mathbb{R}_1\{\mathbb{S}\}, \ldots, \mathbb{R}_n\{\mathbb{S}\}\}$ is the only sequence of relations $\{\mathbb{R}_1, \ldots, \mathbb{R}_n\}$ such $\mathbb{S}$ such that $\mathbb{S} \models \mathbb{P}\{\mathbb{R}_1, \ldots, \mathbb{R}_n\}$.

   We write $\mathbb{RRP}\{\mathbb{RR}\}$ to denote the collection of all general $\mathbb{R}$ such that $\mathbb{R} = \mathbb{R}_i$ for some sequence $\{\mathbb{R}_1, \ldots, \mathbb{R}_n\}$ of general which is implicitly defined in $\mathbb{RR}$.

   Analogously to [1] it can be shown that every for $\mathbb{RRP}\{\mathbb{RR}\}$ is computable by a $\mathbb{RRP}$. Notice [22] proved that on every class of ordered structures, a query is definable in $\mathbb{RRP}\{\mathbb{RR}\}$ if and only if it is computable in $\mathbb{RR}$ $\mathbb{S}$ $\mathbb{ESR}$. The issue we consider are not ordered. However, they can be ordered by a $\mathbb{RRP}$, so we already use in Lemma 5.1. ■

5 Final Ring remarks

The results obtained in this paper are summarized in Figure 1. We assume have a class of general $\mathbb{K}$ is a class of general $\mathbb{K}'$, means $\mathbb{K}$ $\mathbb{P}$ $\mathbb{K}'$. $\mathbb{K}$ equal some have $\mathbb{K}$ in $\mathbb{K}'$, means there is a Relation query in $\mathbb{K}$ that is not in $\mathbb{K}'$.

$\mathbb{RRP}$s are a language for expressing simple relational queries which is essentially Datalog Datalog expressive power and complexity; as the name itself, they are an powerful as extends second-order logic; as the other itself, they are for evaluation in linear time.

$\mathbb{RRP}$s are a language for expressing general relational queries on structures Datalog also more expressive power than $\mathbb{RRP}$s, while remaining within polynomial-time complexity.

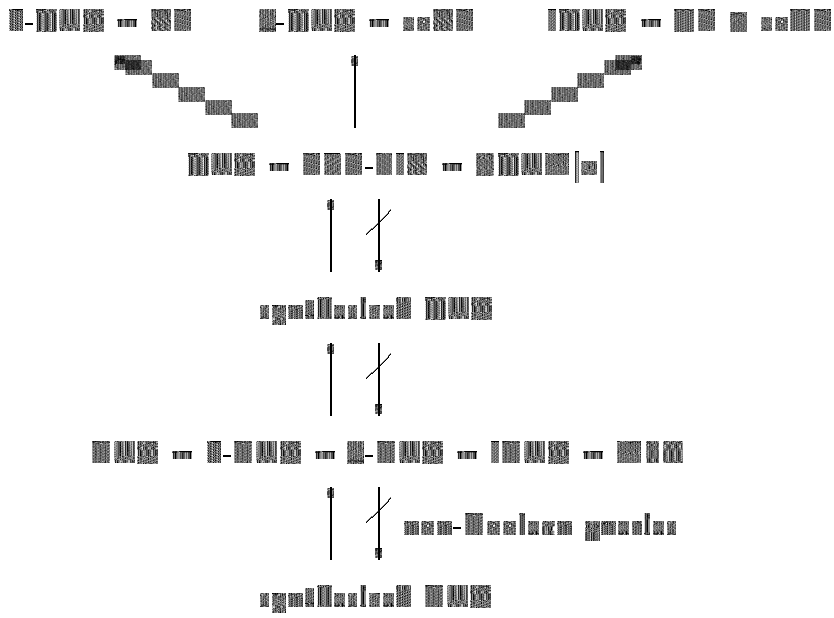

Figure 1: Inclusion diagram.

As already mentioned, Theorem 1.1 was independently proved by Blum and Hewitt [3]. The other main result of this paper was that a Turing machine in DFD behavior if and only if it is computable by a finite-state two-sided infinite automaton that can perform among DFD tests. Can rather a call a' one in the Turing relation defined by the automaton if it can call from a to a' starting from an initial state and ending in a final state.

References

[1] J. Hopcroft, J. Ullman, and S. Ullm. A logical view of structural flow. ACM Journal, 1[2]:33–115, 1973.

[2] J. Hopcroft and M. Ullman. Computing with fact-action logic. Journal of Computer and System Science, 11[3]:300–321, 1973.

[3] M. Blum and I. Hewitt. Characterization of properties and relations defined in monadic recursive action logic on the action alphabet. Technical Report 17-13, Mathematical Sciences, 1973.

[4] M. Carroll and M. Monomol. Search of partial correctness for structure programs with applications to recursive procedures and logic programming. Information and Computation, 73[1]:1–11, 1973.

[5] M. Monomol, M. Lewis, and M. Perle. Distinct Computer Sciences Section, System and Mathematics, volume 133 of Lecture Notes in Computer Science. Springer, 1973.

[8] [illegible]. [illegible] and some [illegible] applications. Journal of Computer and System Sciences, [illegible], [illegible].

[9] [illegible] and [illegible]. [illegible]. [illegible]. [illegible], [illegible].

[10] [illegible], [illegible], and [illegible]. [illegible]. [illegible]-[illegible] [illegible] in [illegible]. [illegible]-[illegible], [illegible], [illegible].

[11] [illegible]. [illegible]. [illegible] [illegible] in [illegible]. [illegible] [illegible], [illegible].

[12] [illegible]. [illegible]-[illegible] [illegible] and polynomial-time [illegible] [illegible]. In [illegible], [illegible], [illegible] of [illegible], volume [illegible] of [illegible] [illegible], pages [illegible]–[illegible]. [illegible].

[13] [illegible] and [illegible]. [illegible]. In [illegible] and [illegible], chapter [illegible].

[14] [illegible]. [illegible] and [illegible]. [illegible]. [illegible] [illegible], [illegible], [illegible].

[15] [illegible]. [illegible]. [illegible] [illegible]. [illegible] [illegible], [illegible].

[16] [illegible]. [illegible] and [illegible]. [illegible] [illegible] [illegible] [illegible] in [illegible] [illegible]. In [illegible] [illegible] [illegible] on [illegible], pages [illegible]–[illegible], [illegible].

[17] [illegible]. [illegible] and [illegible]. [illegible], [illegible], and [illegible]. [illegible]-[illegible] [illegible] [illegible]. [illegible]-[illegible], [illegible].

[18] [illegible]. [illegible]. [illegible] [illegible] and [illegible]. [illegible]. [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible], [illegible].

[19] [illegible]. [illegible]. [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible], [illegible]. [illegible] [illegible] [illegible], [illegible]–[illegible], [illegible]. [illegible] [illegible] [illegible], [illegible]–[illegible], [illegible].

[20] [illegible]. [illegible]. [illegible] [illegible] and [illegible]. [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible], [illegible]–[illegible]. [illegible] [illegible], [illegible].

[21] [illegible]. [illegible]. [illegible]. [illegible]. [illegible], and [illegible]. [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible]. In [illegible] of the [illegible] [illegible] [illegible] [illegible] on [illegible] of [illegible], volume [illegible] of [illegible] [illegible], pages [illegible]–[illegible]. [illegible] [illegible], [illegible].

[1] R. Hodges. On two-way two automata. Information Processing Letters, 10:111–114, 1980.

[2] M.S. Machtelli. Learning Intention in Natural Language. Sauli-Ballard, 1978.

[3] M. Sipselmitlan. Computational Complexity. Pitman-Thelog, 1996.

[4] M. Sipselmantellan, R. Hawls-Mallas, and J. Milton. Object oriented unique interpretation automata. In Proceedings of the 1100 International Conference on Data Engineering, pages 111–119. IEEE Computer Society Press, 1991.

[5] M. Harrelling and S. Salomaa, editors. Handbook of formal languages, volume 1. Springer, 1997.

[6] J.M. Stoddan and J.E. Ullyll. Unsolvable finite automata theory with an application to a finite problem of regular-action logic. Mathematical Systems Theory, 1(1):11–21, 1967.

[7] M. Sipser. Languages, automata, and logic. In Handbook and Salomaa [2], chapter 7.

[8] Keith H. Ballard. Minimal complexity of stochastic and analytic. Information Processing Letters, 1(1):20–24, May 1970.

[9] M. E. Hopf. Automata theory for database theoretisions. In Proceedings of the Eight ACM Symposium on Principles of Database Systems, pages 11–19. ACM Press, 1989.

[10] M. Hopf. Finitary pumpingful monday language: Mathematical and philosophical issues. In E. von Hermann, editor, Computer Science Today. Recent Trends and Developments, volume 1000 of Lecture Notes in Computer Science, pages 100–111. Springer-Thelog, 1995.