A Tableaux-Based Decision Procedure for Multi-Parameter Propositional Schemata

David Cerna *

Technical University of Vienna **

Abstract The class of *regular propositional schemata*, discovered by Aravantinos et al. [4], is a major advancement towards more expressive classes of inductive theorems with a decidable satisfiability problem. Though more expressive than previously known decidable classes outlined by Kapur & Giesl[17], it still requires the burdensome restriction of induction with only one free parameter. In general, unrestricted usage of multiple free parameters in schematic formulae is undecidable for satisfiability [2]. In later work, Aravantinos et al. [6] introduced *normalized clause sets* which have a decision procedure for satisfiability and allow for restricted usage of multiple parameters. In our work, we investigate classes of propositional schemata which allow for multiple free parameters and are more expressive than regular schemata. Specifically, the classes we investigate have a decision procedure for satisfiability testing without requiring the additional theoretical machinery of normalized clause sets. Thus, allowing one to avoid conversion to CNF formulae. Both of the classes we introduce, *linked schemata* and *pure overlap schemata* use the machinery introduced in the earlier works of Aravantinos et al.[4] with only a slight change to the decision procedure.

1 Introduction

The concept of schema has been pervasive throughout the history of logic [14]. First-order Peano arithmetic's usage of an induction schema is a well known example of schema in mathematical logic [22]. There are many other less known examples where schemata were used in both propositional and first-order logic to attain proof theoretic results. For example, results pertaining to proof length, unification, construction of 'proof skeletons', and first-order schematic Hilbert-type systems [7,9,21,19,20]. Also, in the analysis of the Fürstenberg's proof of the infinitude of primes [8], cut elimination resulted in a schema of proofs where the free parameter indexed the number of prime numbers. Very recently, work has been done on schematizing cut-elimination so that an arbitrary number of cuts can be eliminated without instantiating the free parameter of the proof [15].

The usage of schemata that we will focus on for the majority of this paper is schemata as an object level construction iterating propositional formulae. This

* E-mail:cernadavid1@logic.at, Website: http://www.logic.at/people/cernadavid1/

** This work was funded by the Vienna PhD School of Informatics

work was pioneered by Aravantinos et al. [4] with application to the field of repeated circuit verification. This construction has also resulted in discoveries in the field of inductive theorem proving, namely, construction of classes of inductively defined formulae which have a decidable satisfiability problem and are more expressive than those currently known in the field [4,17,18]. Namely, theses classes are *bounded-linear schemata*, *regular schemata*, *nested regular schemata*, and a multiple free parameter generalization of regular schemata through normalized clause set representation, [6]. Decidable classes of inductive theorems discovered by Kapur et al. [17,18], the most prominent work in this area, were mainly universally quantified. Propositional schemata can express both bounded existential and universal quantification through $\wedge$ and $\vee$ iterations.

Our main results are formalizations of the class of linked schemata and pure overlap schemata, both being multiple free parameter extension of regular schema, together with a decision procedure for both classes. This decision procedure is a simple extension of the ST procedure for STAB[4]. Our decision procedure allows one to avoid the conversion of propositional schemata to normalized clause sets [6]. Our work is of a similar vein as Gentzen's work [16] in which he provided a method to fuse multiple inductions together in Peano arithmetic.

Though both classes of schemata we introduce are subclasses of the schemata representable by normalized clause sets and benefit from the satisfiability procedure of normalized clauses sets [6], the existence of a tableaux-based decision procedure for satisfiability testing for these classes remained an open problem. The benefit of a tableaux-based decision procedure is that one does not need to convert the propositional schemata into CNF form to test satisfiability. Note that, if one wants to keep logical equivalence between the original formula and the CNF form, the conversion can result in an exponential increase in formula size.

In this paper, we consider multiple regular schemata (each with its own parameter) such that the propositional symbols of one schema are not found in the other regular schemata. When this property holds, we can use the parts (i.e. the iterations and propositional variables not found in the iterations) to construct a formula with multiple free parameters–we refer to this class as the class of linked schemata. Essentially, we build formulae using the pieces of several regular schemata. Although, this idea is quite simple, it provides a class of schemata extending regular schemata which still has a tableaux-based decision procedure for satisfiability.

Next we investigate when it is possible for the propositional symbols to occur in two or more *linked* regular schemata, i.e. the same propositional symbol has occurrences indexed by two different parameters. To answer this question, we develop the concept of relative pure literals, literals which are pure when considering occurrences indexed by another parameter. This concept is used to construct the class of pure overlap schemata.

Both linked and pure overlap schemata are extensions of regular schemata, but after applying several tableaux extension rules to the constructed tableau, It is possible to reduce the branches of the constructed tableau to tableaux

branches which are decidable using the decision procedure for regular schemata. Essentially, they are both propositional extensions of the class. It is not completely clear if these classes of schemata are the most expressive classes such that their satisfiability problem can be reduced to the satisfiability problem for regular schemata. An open problem regarding this point is whether the purity constraint can be relaxed and retain the reduction– results of Aravantinos et al. [4] (Thm. 6.2) suggests that this is not going to be the case.

Overall, our paper provides a simpler and more natural alternative to normalized clause set representation when deciding satisfiability for certain classes of multiple-parameter schemata.

The rest of this paper is structured as follows, Sec. 2 will be necessary background material from Aravantinos et al. [4], in Sec. 3 we formalize the construction of linked schemata, in Sec. 4 we formalize the construction of pure overlap schemata , in Sec. 5 we provide a decision procedure for the satisfiability problem of pure overlap schemata. Finally, in Sec. 6 we conclude the paper and shortly discuss the open problems.

2 Background

2.1 Propositional Schemata

The indexing language for standard schematic propositional logic as considered in Aravantinos et al. [4] is the set of *linear arithmetic terms* (denoted by $\mathcal{Z}$) built using the language $\{0, s(\cdot), +, -\}$ and a countably infinite set of variables $\mathcal{V}$. Multiplication is considered as a shorthand for terms of the form $x{+}x{+}x{+}x = 4{\cdot}x$ and is not a real operator in the language, nor is it a necessary one. To stick to the framework of Aravantinos et al. [4] $\mathbb{Z}$ is considered as the standard model of the terms in $\mathcal{Z}$.

Definition 1 (Indexed Proposition[4]). *Let $\mathcal{P}$ be a fixed and countably infinite set of propositional symbols. An* indexed proposition *is an expression of the form $p_{\mathbf{a}}$ where $p \in \mathcal{P}$ and $\mathbf{a} \in \mathcal{Z}$. An indexed proposition $p_{\mathbf{a}}$ s.t. $\mathbf{a} \in \mathbb{Z}$ is called a propositional variable.*

Definition 2 (Formula Schemata[4]). *The set of* formula schemata *is the smallest set satisfying the following properties.*

- $\bot, \top$ *are formula schemata.*
- *If $\mathbf{a}, \mathbf{b} \in \mathcal{Z}$ then $\mathbf{a} < \mathbf{b}$ is a formula schema.*
- *Each indexed proposition is a formula schema.*
- *If ϕ_1, ϕ_2 are formula schemata then $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\neg\phi_1$ are formula schemata.*
- *If ϕ is a formula schema not containing $<$, and if $\mathbf{a}, \mathbf{b} \in \mathcal{Z}$, where i is an arithmetic variable, then $\bigwedge_{i=\mathbf{a}}^{\mathbf{b}} \phi$, $\bigvee_{i=\mathbf{a}}^{\mathbf{b}} \phi$ are formula schemata.*

Example 1. Consider the formula:

$$\varphi = q_1 \wedge \bigwedge_{i=0}^{n} \left(p_{i+2n} \wedge \left(\bigvee_{j=n}^{2n+1} \neg q_{n-j} \vee q_{j+1} \right) \right) \wedge 0 \leq n$$

φ is a formula schema.

Formula schemata are inherently finite. We will label the indexed propositions, $\top$, $\bot$ and statements of the form $a < b$, as *atoms*. Formula schemata of the form $\bigwedge_{i=\mathbf{a}}^{\mathbf{b}} \phi$ and $\bigvee_{i=\mathbf{a}}^{\mathbf{b}} \phi$ will be called *iterations*. A formula schema whose constituents are any of the following: $\top$, $\bot$, and $a < b$, is an *arithmetic formula*. Also, it is taken as a standard that arithmetic formulae of the form $a < b$ can only occur outside of iterations. This constraint is necessary being that $a < b$ is interpreted as an iteration, i.e.

$$a < b \equiv \bigvee_{i=a+1}^{b} \top \tag{1}$$

Also, we use $a = b$ as an abbreviation for $\neg(b < a) \wedge \neg(a < b)$ and $a \leq b$ as an abbreviation for $\neg(b < a)$. Iterations have both *free* and *bound* variables, where free variable and *parameter* are synonymous. A bound variable i is a variable in the scope of an iteration $\mathit{\Pi}_{i=a}^{b} \phi_i$ where $\mathit{\Pi} = \{\bigvee, \bigwedge\}$. A *substitution* is a function mapping all the free variables to linear expressions. If a substitution σ is applied to a schema φ, i.e $\varphi\sigma$ such that the domain of σ is every free variable in φ, then the linear expressions of φ are integer terms, i.e. all indices in φ are variable free.

Definition 3 (Interpretation [4]). *An* interpretation of the schematic language *is a function mapping every parameter to an integer and every propositional variable to a truth value* $\boldsymbol{T}$ *or* $\boldsymbol{F}$*. The substitution and interpretation will be denoted as* σ *and* $\mathcal{I}$*, respectively.*

Example 2. An Interpretation $\mathcal{I}$ such that φ from Ex. 1 is modelled by $\mathcal{I}$ would be $\sigma \equiv \{n \leftarrow 0\}$ and $q_1 = \mathbf{T}, p_0 = \mathbf{T}, q_0 = \mathbf{T}, q_{-1} = \mathbf{T}, q_1 = \mathbf{F}, q_2 = \mathbf{T}$

Definition 4 (Semantics of Schematic Formulae [4]). *The semantics of a schematic formula* φ *in a given interpretation* I*, denoted by* $[\![\varphi]\!]_{\mathcal{I}}$*, is defined as follows:*

- $[\![\top]\!]_{\mathcal{I}} = \mathbf{T}$ *and* $[\![\bot]\!]_{\mathcal{I}} = \mathbf{F}$
- $[\![\mathbf{a} < \mathbf{b}]\!]_{\mathcal{I}} = \mathbf{T} \Leftrightarrow [\![\mathbf{a}]\!]_{\mathcal{I}} <_{\mathbb{Z}} [\![\mathbf{b}]\!]_{\mathcal{I}}$
- $[\![P_{\mathbf{a}}]\!]_{\mathcal{I}} = \mathcal{I}(P_{[\![\mathbf{a}]\!]_{\mathcal{I}}})$ *for* $P \in \mathcal{P}$
- $[\![\neg\varphi]\!]_{\mathcal{I}} = \mathbf{T} \Leftrightarrow [\![\varphi]\!]_{\mathcal{I}} = \mathbf{F}$
- $[\![\varphi \vee \psi]\!]_{\mathcal{I}} = \mathbf{T} \Leftrightarrow [\![\varphi]\!]_{\mathcal{I}} = \mathbf{T}$ *or* $[\![\psi]\!]_{\mathcal{I}} = \mathbf{T}$
- $[\![\varphi \wedge \psi]\!]_{\mathcal{I}} = \mathbf{T} \Leftrightarrow [\![\varphi]\!]_{\mathcal{I}} = \mathbf{T}$ *and* $[\![\psi]\!]_{\mathcal{I}} = \mathbf{T}$
- $[\![\bigvee_{i=\mathbf{a}}^{\mathbf{b}} \varphi_i]\!]_{\mathcal{I}} = \mathbf{T} \Leftrightarrow \exists \alpha \in \mathbb{Z}$ *such that* $[\![\mathbf{a}]\!]_{\mathcal{I}} \leq_{\mathbb{Z}} \alpha \leq_{\mathbb{Z}} [\![\mathbf{b}]\!]_{\mathcal{I}}$ *and* $[\![\varphi_i]\!]_{\mathcal{I}[\alpha/i]} = \mathbf{T}$
- $[\![\bigwedge_{i=\mathbf{a}}^{\mathbf{b}} \varphi_i]\!]_{\mathcal{I}} = \mathbf{T} \Leftrightarrow \forall \alpha \in \mathbb{Z}$, $[\![\mathbf{a}]\!]_{\mathcal{I}} \leq_{\mathbb{Z}} \alpha \leq_{\mathbb{Z}} [\![\mathbf{b}]\!]_{\mathcal{I}}$ *implies* $[\![\varphi_i]\!]_{\mathcal{I}[\alpha/i]} = \mathbf{T}$

In the above definition, by $[\![\varphi_i]\!]_{\mathcal{I}[\alpha/i]}$ we mean every occurrence of i in φ_i is replaced by α. A propositional schema φ is *valid* (respectively *satisfiable*) iff for all (exists an interpretation) interpretations $\mathcal{I}$ s.t. $[\![\varphi]\!]_{\mathcal{I}} = T$. $\mathcal{I}$ is called a *model* of φ, written as $\mathcal{I} \models \varphi$. Two schemata φ, ψ are equivalent (written $\varphi \equiv \psi$) iff $\mathcal{I} \models \varphi \Leftrightarrow \mathcal{I} \models \psi$. φ and ψ are sat-equivalent (written $\varphi \equiv_S \psi$) iff φ and ψ are both satisfiable or both unsatisfiable (not necessarily by the same model).

Definition 5 (Unrolling Iterations [4]). *The following set S of rewrite rules is used to unroll the iterations of a given schematic formula φ:*

$$S = \begin{cases} \bigvee_{i=\mathbf{a}}^{\mathbf{b}} \psi \longrightarrow \bot & \mathbf{a}, \mathbf{b} \in \mathbb{Z} \text{ and } \mathbf{b} <_{\mathbb{Z}} \mathbf{a} \\ \bigwedge_{i=\mathbf{a}}^{\mathbf{b}} \psi \longrightarrow \top & \mathbf{a}, \mathbf{b} \in \mathbb{Z} \text{ and } \mathbf{b} <_{\mathbb{Z}} \mathbf{a} \\ \bigvee_{i=\mathbf{a}}^{\mathbf{b}} \psi \longrightarrow \left(\bigvee_{i=\mathbf{a}}^{\mathbf{b}\dot{-}1} \psi\right) \vee \psi\,[\mathbf{b}/i] & \mathbf{a}, \mathbf{b} \in \mathbb{Z} \text{ and } \mathbf{a} \leq_{\mathbb{Z}} \mathbf{b} \\ \bigwedge_{i=\mathbf{a}}^{\mathbf{b}} \psi \longrightarrow \left(\bigwedge_{i=\mathbf{a}}^{\mathbf{b}\dot{-}1} \psi\right) \wedge \psi\,[\mathbf{b}/i] & \mathbf{a}, \mathbf{b} \in \mathbb{Z} \text{ and } \mathbf{a} \leq_{\mathbb{Z}} \mathbf{b} \end{cases} \tag{2}$$

By $\leq_{\mathbb{Z}}$ we are referring to the standard ordering over the integers.

Definition 6 (Regular Schemata (as written in [4])). *A propositional schema ϕ is* regular *if it has a unique parameter n and if it is* flat*, of* bounded propagation *and* aligned *on $[\alpha, n-\beta]$:*

1) *A schema is* flat *if every $\Pi_{i=a}^{b}\psi$ occurring in the schema ψ does not contain an iteration, , where $\Pi \in \{\bigvee, \bigwedge\}$.*
2) *A schema is of* bounded propagation *if every atom that occurs in an iteration $\Pi_{i=a}^{b}\psi$ is of the form $P_{i+\gamma}$ for some $\gamma \in \mathbb{Z}$, where $\Pi \in \{\bigvee, \bigwedge\}$.*
3) *A schema is* aligned *on $[c, d]$ if all iterations occurring in the schema are of the form $\Pi_{i=c}^{d}\psi$, where $\Pi \in \{\bigvee, \bigwedge\}$.*

Example 3. Consider the following schema:

$$\varphi = p_0 \wedge \left(\bigwedge_{i=0}^{n} \neg p_i \vee p_{i+1}\right) \wedge \neg p_n \wedge 0 \leq n \tag{3}$$

φ is a regular schemata.

2.2 Basics of STAB and the ST procedure

We now overview the main ingredients of the ST decision procedure of the STAB framework introduced in Aravantinos et al. [4]. In this paper, we only rely on the existence of the ST procedure and the propositional tableaux extension rules to define an extended decision procedure for our newly defined classes of schemata.

Definition 7 (Tableau). *A* tableau *is a tree T s.t. each node N occurring in T is labelled by a set of schemata written $\Phi_T(N)$.*

Definition 8 (Extension Rules). *The* extension rules *of the STAB procedure are as follows:*

Propositional Rules

- $\varphi \wedge \psi \Rightarrow \varphi, \psi$
- $\varphi \vee \psi \Rightarrow \varphi \mid \psi$

Iteration Rules

- $\bigwedge_{i=a}^{b} \varphi \Rightarrow a \leq b, \varphi\,[b/i] \wedge \bigwedge_{i=a}^{b-1} \varphi \;\Big|\; b < a$
- $\bigvee_{i=a}^{b} \varphi \Rightarrow a \leq b, \varphi\,[b/i] \vee \bigvee_{i=a}^{b-1} \varphi$

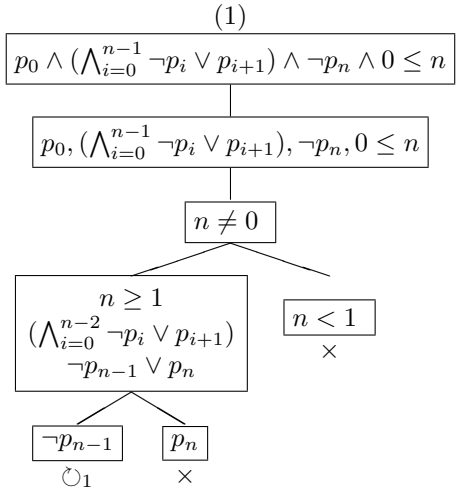
Closure Rule

- $p_a, \neg p_b \Rightarrow p_a, \neg p_b, a \neq b$

The way the STAB extension rules work is by extending currently constructed tableau with new leaves containing all the formulae of the prior node minus the formula φ on which the extension rule was applied. The parts of φ will be added to the leaves in accordance with the extension rule definitions. The symbol | in the extension rules means that the constructed tableau branches when this rule is applied. The closure rule, rather than extending the constructed tableau, tells us that there is no need to extend the considered branch because it contains an unsatisfiable sub-branch.

Theorem 1. *There is a decision procedure for satisfiability testing of regular schemata (ST procedure) based on the STAB extension rules (Def. 8) and an additional rule to deal with looping, which terminates on every regular schema. The procedure is sound and complete for regular schemata.*

Example 4. We provide an example of the ST procedure producing a closed tableau for the regular schema of example 3. Note that not every available formula is passed down the constructed tableau in the diagram.



The symbol $\circlearrowleft_1$ at the bottom of the left-most branch represents the looping rule. Essentially it means that the branch at the denoted point is the same as the branch at (1) (the top of the tableau), but for $n-1$ instead of n. We will not delve deeper into the theory behind the looping rule as we only rely on the existence of such a rule for our procedure to work– for more details on the looping we refer to [4].

Finally, we recall the following two concepts over a set Φ of schemata: the interval constraints ($IC(\Phi)$) and the conjunction of arithmetic formulae in Φ ($\Phi_{\mathcal{Z}}$). The formula $IC(\Phi)$ is the conjunction of the arithmetic formulae $min_\phi(i) \leq i \wedge i \leq max_\phi(i)$ for each $\phi \in \Phi$ and for each bound variable in ϕ. We assume that all bound variables are distinct in Φ and $min_\phi(i)$ is defined as the minimal value that can be assigned to the bound variable i, whereas $max_\phi(i)$ is the maximum value that can be assigned to the bound variable i.

Definition 9 (Pure Literal). *A literal p_a (respectively $\neg p_a$) is* pure *in a set of schemata Φ iff for every occurrence of a literal $\neg p_b$ (respectively p_b) in Φ, the arithmetic formula $\Phi_{\mathcal{Z}} \wedge IC(\Phi) \wedge a = b$ is unsatisfiable.*

This definition will be modified to formalize the class of pure overlap schemata.

3 Linked Schemata

The class of linked schemata is an extension of regular schemata based on the following observation:

$$\left(\bigwedge_{i=1}^{n} p_i\right) \wedge \left(\bigvee_{j=n+1}^{m} \neg p_i\right) \equiv_S \left(\bigwedge_{i=1}^{n} p_i\right) \wedge \left(\bigvee_{j=1}^{m} \neg q_i\right). \tag{4}$$

Simply, we choose the interpretations such that $[\![p_{n+k}]\!]_{\mathcal{I}} = [\![q_k]\!]_{\mathcal{I}}$ for $k \in [1, m]$. By the finiteness of the language, we can separate the integers into two distinct parts, those greater than n and those less than n. Thus, the propositional variable p in the interval $[1, n]$ is invariant to the labelling of the propositional variable in the interval $[n+1, m]$. They can share the same name or not, the assignment will not influence the interpretations which model the schema. This observation is similar to the reduction from monadic predicate logic with monadic function symbols to monadic predicate logic without monadic function symbols, as outlined in Sec. 6.2 of "The Classical Decision Problem"[10].

3.1 Construction

The simplest way to understand the construction of the class of linked schemata is that any regular schema consists of atoms (specifically, ones not contained in iterations) and iterations. We will refer to these "parts" as the *principal objects*, denoted by $\mathcal{P}(\phi)$ of a schema ϕ. We consider sets Φ of regular schemata, such that the propositional symbols are distinct with regards to the regular schemata in

the set, i.e. if $\phi, \psi \in \Phi$ and ϕ contains a propositional variable using the symbol p then ψ cannot contain propositional variables using this symbol. We can compute $\bigcup_{\phi \in \Phi} \mathcal{P}(\phi)$ without any propositional symbols occuring in two iterations indexed by different free parameters. Using this set of "parts" and the propositional connectives $\neg$, $\vee$, and $\wedge$ we can construct new formulae. The rest of this section will be focused on the formalization of this concept.

Definition 10. *Let $p \in \mathcal{P}$ be a propositional symbol and φ a formula schema, then $occ(p, \varphi) = 1$ iff p occurs in φ, otherwise it is $occ(p, \varphi) = 0$*

Definition 11 (principal Objects). *Given a schema φ we can construct the set of* principal objects *$\mathcal{P}(\varphi)$ using the following inductive definition:*

- $\mathcal{P}(P_a) \Rightarrow \{P_a\}$
- $\mathcal{P}(\bigvee_{i=a}^{b} \psi) \Rightarrow \left\{\bigvee_{i=a}^{b} \psi\right\}$
- $\mathcal{P}(\bigwedge_{i=a}^{b} \psi) \Rightarrow \left\{\bigwedge_{i=a}^{b} \psi\right\}$
- $\mathcal{P}(\neg\psi) \Rightarrow \mathcal{P}(\psi)$
- $\mathcal{P}(\phi \vee \psi) \Rightarrow \mathcal{P}(\phi) \cup \mathcal{P}(\psi)$
- $\mathcal{P}(\phi \wedge \psi) \Rightarrow \mathcal{P}(\phi) \cup \mathcal{P}(\psi)$

One can consider $\mathcal{P}(\varphi)$ as a specially constructed set of formula schema.

Example 5. Let use compute the set of principal objects of the following regular schema:

$$\varphi \equiv (0 \leq n) \wedge P_0 \wedge \bigwedge_{i=1}^{n} (\neg P_{i-1} \vee P_i) \wedge \neg P_n \tag{5}$$

We get $\mathcal{P}(\varphi) = \{(0 \leq n), P_0, \bigwedge_{i=1}^{n} (\neg P_{i-1} \vee P_i), P_n\}$

We will abbreviate the set of propositional connectives used as $\mathcal{O} = \{\wedge, \vee, \neg\}$. By $\psi \in cl_{\mathcal{O}}(\Phi)$, we mean that ψ can be constructed using the set of formula schema Φ and the logical connective set $\mathcal{O}$.

Example 6. Using the principal object set from Ex. 5 and the set of operators $\mathcal{O} = \{\wedge, \vee, \neg\}$, some of the formulae we can construct are:

$$\psi_1 = (0 \leq n) \wedge P_0 \wedge \bigwedge_{i=1}^{n} (\neg P_{i-1} \vee P_i) \wedge \neg P_n \tag{6}$$

$$\psi_2 = \big((0 \leq n) \wedge P_0 \wedge \textstyle\bigwedge_{i=1}^{n} (\neg P_{i-1} \vee P_i) \wedge \neg P_n\big) \vee \big((0 \leq n) \wedge \neg P_0 \wedge \textstyle\bigwedge_{i=1}^{n} (\neg P_{i-1} \vee P_i) \wedge P_n\big) \tag{7}$$

It is not necessary that the constructed formulae are valid, satisfiable, or unsatisfiable. One can check that both $\psi_1, \psi_2 \in cl_{\mathcal{O}}(\mathcal{P}(\varphi))$.

Lemma 1. *If φ is a regular schema, then all $\psi \in cl_{\mathcal{O}}(\mathcal{P}(\varphi))$ have the same aligned interval as φ.*

Proof. Assuming that φ has an aligned interval $[\alpha, n-\beta]$, then any, of its parts must have an aligned interval of at most $[\alpha, n-\beta]$ and are themselves regular schema. Thus, ψ is a boolean combination with the same aligned interval, implying that its aligned interval must be the same. □

Using this simple result we will define the class of linked schemata, as follows.

Definition 12 (The class of Linked Schemata). *Let us consider the class Λ of all finite sets Φ of regular schemata such that for all propositional symbols p, we have that $\left(\sum_{\phi\in\Phi} occ(p,\phi)\right)$ is either 1 or 0, we define the class* **LS** *of* linked schemata *as*

$$\mathbf{LS} = \bigcup_{\Phi\in\Lambda} cl_{\mathcal{O}}\left(\bigcup_{\phi\in\Phi}\mathcal{P}(\phi)\right)$$

Lemma 2. *If φ is a regular schema, then it is a linked schema.*

Proof. By definition 12, we can consider the set $\Phi = \{\varphi\}$, also, $\varphi \in cl_{\mathcal{O}}(\mathcal{P}(\varphi))$, and thus, $\varphi \in \mathbf{LS}$.
□

Theorem 2. *The class of regular schemata is contained but not equal to the class of linked schemata.*

Proof. We prove this by providing an example, see Ex. 7, of a linked schema which is not a regular schema.
□

Example 7. Let us consider Φ containing the following three regular schemata. In what follows, we write $A \leftrightarrow B$ as an abbreviation for $(\neg A \vee B) \wedge (\neg B \vee A)$:

$$\varphi_1 = \bigvee_{i=1}^{k} \neg P_i \wedge \neg \bigvee_{i=1}^{k} \neg P_i \tag{8a}$$

$$\varphi_2 = \bigvee_{i=1}^{m} Q_i \wedge \bigvee_{i=1}^{m} R_i \wedge \bigwedge_{i=1}^{m} Q_i \leftrightarrow R_i \tag{8b}$$

$$\varphi_3 = \bigwedge_{i=1}^{n} M_i \tag{8c}$$

We can construct the following **LS** formula using Φ:

$$\left(\left(\bigvee_{i=1}^{k}\neg P_i \rightarrow \bigvee_{i=1}^{m} Q_i\right) \wedge \left(\bigvee_{i=1}^{m} R_i \rightarrow \bigwedge_{i=1}^{n} M_i\right) \wedge \bigwedge_{i=1}^{m}(Q_i \leftrightarrow R_i)\right) \rightarrow \left(\bigvee_{i=1}^{k}\neg P_i \rightarrow \bigwedge_{i=1}^{n} M_i\right). \tag{9}$$

Formula 9 gives a formalization of the composition of certain boolean functions when one function's range has the same number of bits as another function's domain. This formula is obviously not regular, but it is linked. This concludes the proof of Thm. 2.

4 Pure Overlap Schemata

In this section we show how one can weaken the restriction that propositional symbols occur indexed by only one parameter. Consider the following formula schema ψ:

$$0 \leq n \wedge \left(\bigwedge_{i=0}^{n} p_i\right) \vee \left(\bigwedge_{i=0}^{m} \neg p_i\right) \wedge 0 \leq m \tag{10}$$

It is not a linked schema because p occurs indexed by two different parameters, however, using the tableaux extension rule for propositional $\vee$ we see that the occurrences are handled by two different branches, thus each parameter can be handled separately. It is also important to note that

$$0 \leq n \wedge \left(\bigwedge_{i=0}^{n} p_i\right) \vee \left(\bigwedge_{i=0}^{m} \neg q_i\right) \wedge 0 \leq m \tag{11}$$

Replacing p with q in Eqn. 10 results in Eqn. 11, which changes the formula from valid to satisfiable (only when $0 \leq n, m$). Thus, we cannot reduce this formula to linked schemata without changing its semantic properties. To deal with this problem we introduce relatively pure literals, based on the observation that if the negation of a literal occurs in the same branch indexed by a different parameter then the literal must not be of arithmetic importance. We then show that relatively pure literals can be dropped without effecting satisfiability of the considered pure overlap schemata.

4.1 Construction

We first introduce the notion of relatively pure literals and detail the construction of pure overlap schemata.

Definition 13 (Iteration Invariant DNF (IIDNF)). *The* Iteration Invariant disjunctive normal *form of a linked schema is a schema of the form:*

$$(\varphi_{1,1} \wedge \cdots \wedge \varphi_{1,n_1}) \vee \cdots \vee (\varphi_{m,1} \wedge \cdots \wedge \varphi_{m,n_m})$$

where $m, n_1, \cdots, n_m \in \mathbb{N}$ (note they are not free parameters, but rather meta variables) and $\varphi_{i,j}$ is either an iteration, an atom, or negated atom. We will refer to the formula $(\varphi_{i,1} \wedge \cdots \wedge \varphi_{i,n_i})$ as clauses C_i for $i \in [1, m]$, That is, given a formula φ in IIDNF we will write $C_i \in \varphi$ as the i^{th} clause of φ.

Lemma 3. *Given a set of regular schemata Φ, for all $\psi \in cl_{\mathcal{O}}\left(\bigcup_{\phi\in\Phi}\mathcal{P}(\phi)\right)$ there exists an IIDNF of ψ.*

Proof. Since, iterations are not unfolded in the creation of an IIDNF form of ψ, the problem reduces to showing that all propositional formulae have a DNF form, which is a well known result. Also, it is possible to put a regular schemata into *Negation Normal Form* (NNF) because negation can be passed over iterations, i.e $\neg\bigwedge_{i=a}^{b}\phi_i \equiv_{\models} \bigvee_{i=a}^{b}\neg\phi_i$ and $\neg\bigvee_{i=a}^{b}\phi_i \equiv_{\models} \bigwedge_{i=a}^{b}\neg\phi_i$.
□

Definition 14 (Relatively Pure Literal). *Given a set of regular schemata Φ, let $\psi \in cl_{\mathcal{O}}\left(\bigcup_{\phi\in\Phi}\mathcal{P}(\phi)\right)$ and ψ' be the IIDNF of ψ. A literal p_a ($\neg p_a$) is* relatively pure *in ψ iff for every clause $C \in \psi'$ and for any two distinct regular schemata $\varphi_1, \varphi_2 \in \Phi$ used to construct ψ, where $p_a \in C$ ($\neg p_a \in C$), $\neg p_b \in C$ ($p_b \in C$), $p_a \in \varphi_1$ ($\neg p_a \in \varphi_1$) and $\neg p_b \in \varphi_2$ ($p_b \in \varphi_2$), the arithmetic formula $\Phi_{\mathcal{Z}} \wedge IC(\Phi) \wedge a = b$ is unsatisfiable, where $\Phi = \mathcal{P}(C)$.*

Example 8. Consider the schemata:

$$\neg(5 < n) \wedge \left(\bigwedge_{i=0}^{n} p_i\right) \wedge \left(\bigwedge_{j=6}^{m} \neg p_i\right) \wedge 0 \leq m \tag{12}$$

The literal p_i ($\neg p_i$) is relatively pure in this example.

We will refer to a schema as *relatively pure* if all the literals in the schema are either relatively pure or in the IIDNF of the schema they only occur in clauses being indexed by a single parameter. The non-IIDNF form of a relatively pure schema is also relatively pure. Given a set of regular schemata Φ, let $cl^{rp}_{\mathcal{O}}(\Phi)$ be the set of all schema which can be constructed using the logical connectives $\mathcal{O}$ such that they are relatively pure.

Definition 15 (The class of Pure Overlap Schemata). *Let us consider the class Λ of all finite sets Φ of regular schemata. We define the class of* pure overlap schemata *as*

$$\mathbf{POS} = \bigcup_{\Phi\in\Lambda} cl^{rp}_{\mathcal{O}}\left(\bigcup_{\phi\in\Phi}\mathcal{P}(\phi)\right)$$

It should be noted that even though the definition of relatively pure literals uses the IIDNF of a positional schema it is not the case that members of **POS** must be in IIDNF.

Example 9. Both Ex. 8 and Eqn. 10 are in the class of pure overlap schemata.

Lemma 4. *If φ is a linked schema, then it is a pure overlap schema.*

Proof. A linked schema is a pure overlap schema where each propositional variable is indexed by only one parameter.
□

Theorem 3. *The class of linked schemata is contained but not equal to the class of pure overlap schemata.*

Proof. Eqn. 10 is a pure overlap schema but not a linked schema.
□

5 A Decision procedure for POS

We now introduce a decision procedure for the class POS of schemata, by using and extending results of [4], as follows.

Algorithm 1 ($ST^{\mathbf{POS}}$ Procedure) *Given a schema $\varphi \in$ **POS** in negation normal form. The following algorithm, called the $ST^{\mathbf{POS}}$ procedure, decides the satisfiability of φ:*

1) *Apply STAB propositional extension rules with highest priority until no more can be applied. This results in m sets of atoms and iterations referred to as $B_1, \ldots, B_m$.*
2) *For each B_i, we separate B_i into n (the number of parameters in B_i) sub-branches $B_{(i,1)}, \cdots B_{(i,n)}$, where each $B_{(i,j)}$ contains iterations and atoms indexed by a single parameter. Atoms without a free parameter in the indices can be added to every $B_{(i,j)}$. We will mark such a sub-branching with $\otimes_n$ where n is the number of parameters on the branch.*
3) *Run the ST procedure on the sub-branch $B_{(i,j)}$.*
4) *For any branch B_i, if one of its sub-branches $B_{(i,j)}$ has a closed tableau after following the ST procedure, then the branch B_i is closed.*

Let us make the following observation about the $ST^{\mathbf{POS}}$ decision procedure. When it comes to constructing the interpretation for a formula in **POS** we specifically defined the class such that the procedure to construct the model would be precisely the procedure used for regular schemata, except the number of possible models would increase. For linked schemata this is obvious, the propositional symbols are distinct in every sub-branch. However, for pure overlap schemata two distinct sub-branches (of the same branch) can contain the same propositional symbol, but by Def. 14 the occurrences are distinct from each other arithmetically if one occurrence is negated and the other occurrence is not. Thus, when a propositional symbol occurs on two distinct sub-branches and the two occurrences are not arithmetically distinct, the two occurrences must be of the same polarity. In this case when one sub-branch forces the propositional variable using the positional symbol to be true (false in the case of a negated literal), the other sub-branches will also interpret this literal as true. In some sense one can consider it as a local tautology which can be removed from consideration when constructing the model.

Theorem 4. *The $ST^{\mathbf{POS}}$ procedure terminates for* **POS**.

Proof. The key to the termination is that we only need to decompose the members of **POS** using the procedure outlined above. This decomposition process always terminates being that we are, up to this point, only applying propositional tableaux extension rules. When the formulae are completely decomposed we use the ST procedure on each sub-branch. The procedure is known to terminate for regular schemata [4] and each of the sub-branches is regular. □

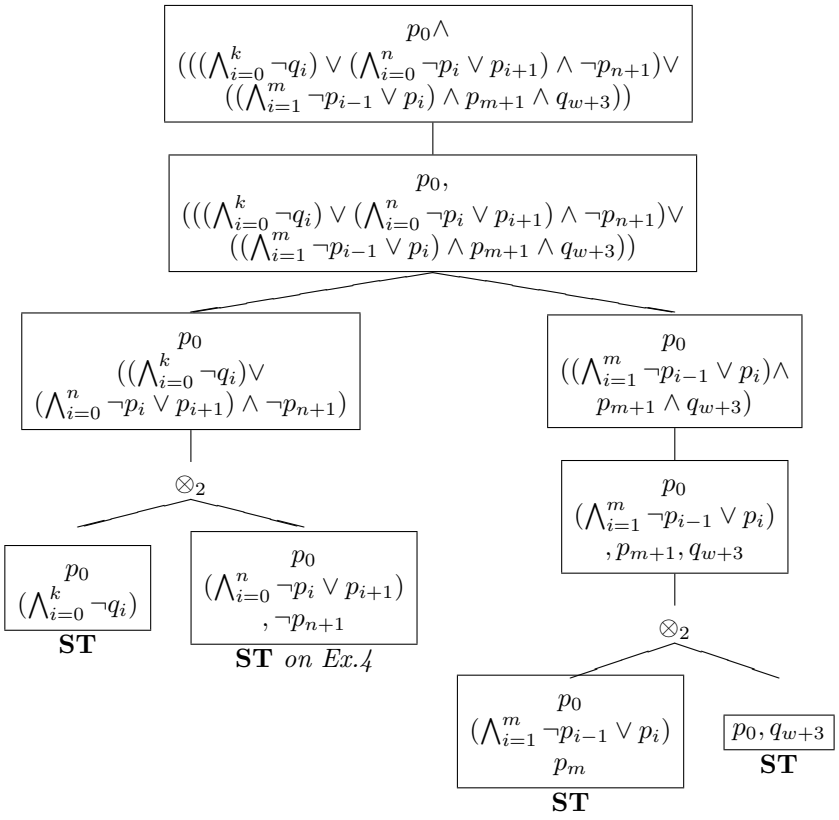
In regards to the soundness and completeness of $ST^{\mathbf{POS}}$ the procedure, it was shown that STAB is sound and complete for all propositional schemata [4] (Sec. 5.4). The propositional schemata we introduce in this paper are constructed using exactly the same language as in the work by Aravantinos et al. [4] Our extension of STAB with the sub-branching rule does not change the soundness and completeness results being that the sub-branching rule, rather than being an additional tableaux rule, is more a method to enforce termination. It essentially states that instead of considering the given branch as a whole we consider it in parts using the same tableaux rules introduced for STAB in prior work.

Theorem 5. *The $ST^{\mathbf{POS}}$ decision procedure is sound and complete for all propositional schemata $\psi \in \mathbf{POS}$.*

Example 10. We conclude this section by illustrating our $ST^{\mathbf{POS}}$ decision procedure on the following formula ψ:

$$p_0 \wedge \Big(\Big(\Big(\bigwedge_{i=0}^{k} \neg q_i \Big) \vee \Big(\bigwedge_{i=0}^{n} \neg p_i \vee p_{i+1} \Big) \wedge \neg p_{n+1} \Big) \vee \Big(\Big(\bigwedge_{i=0}^{m} \neg p_{i-1} \vee p_i \Big) \wedge p_{m+1} \wedge \neg q_{w+3} \Big) \Big)$$

Applying $ST^{\mathbf{POS}}$ on the above formula, we obtain the following branching tree (corresponding to the run of $ST^{\mathbf{POS}}$):



Interesting result of this derivation is that the assignment to w influences the interpretation modelling the formula. If an interpretation $\mathcal{I}$ assigns $q_{-1} = T$,$q_0 = F$,$p_0 = T$,$p_{-1} = F$, $p_5 = F$,$w \leftarrow -4$, $n \leftarrow -2$, $k \leftarrow 0$, and $m \leftarrow 5$ then $\mathcal{I} \models \psi$. But if $\mathcal{I}$ assigns to $n \leftarrow 0$ keeping the same propositional variable assignments, then $\mathcal{I} \not\models \psi$.

6 Conclusion and future work

In this work we have shown that the ST procedure of Aravantinos et al. [4] can be extended to handle more expressive classes of schemata which allow for restricted use of multiple free parameters. The two classes shown, though their construction is awkward, are simple to conceptually understand and work with. Also, neither requires the heavy machinery of normalized clause sets, nor do the classes require a conversion of the schemata into a clausal normal form. Also, the introduced decision procedure $ST^{\mathbf{POS}}$ is sound, complete, and terminates for all propositional schemata in the class **POS**. Though an advantage normalized clause sets have over both of the introduced classes of schemata is that they can handle propositional variables being indexed by multiple free parameters without restriction. This is one of the significant advantages to separating the

propositional part from the equational part, and using a levelled resolution calculus. When it is not required to have unrestricted usage of the propositional variables it suffices to use STAB. This also has the added value of compression being that it is possible for clausal form to result in an exponential increase in the size of the formula.

As for future work, further increase in expressivity by relaxing the purity constraint does not seem feasible as this would require two parameters to be active in the same branch. This is when the undecidability result for propositional schemata [4] stops us in our tracks. However, investigating how the new classes outlined here can interact with the class of *regular nested schemata* [3] could lead to new expressivity results. In particular, we are interested in the relationship between alternation-free μ-calculus [11] and such a class of schemata. Also, Aravantinos et al. [5] investigated the relationship between LTL and regular schemata. Being that pure overlap schemata are a super class of regular schemata it is quite possible that a more expressive temporal logic is related to pure overlap schemata or linked schemata. In either case this work has broaden the scope of application of propositional schemata.

Acknowledgements: I would like to give special thanks to Daniel Weller[1] and Alexander Leitsch[2] for their help with constructing a concise mathematical formalism, as well as Giselle Reis [3] for help with editing of the formalisms.

References

1. Peter Aczel. An introduction to inductive definitions. In Jon Barwise, editor, *HANDBOOK OF MATHEMATICAL LOGIC*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, pages 739 – 782. Elsevier, 1977.
2. Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. A schemata calculus for propositional logic. In Martin Giese and Arild Waaler, editors, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 5607 of *Lecture Notes in Computer Science*, pages 32–46. Springer Berlin Heidelberg, 2009.
3. Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. A decidable class of nested iterated schemata. In *Proceedings of the 5th international conference on Automated Reasoning*, IJCAR'10, pages 293–308, Berlin, Heidelberg, 2010. Springer-Verlag.
4. Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. Decidability and undecidability results for propositional schemata. *J. Artif. Int. Res.*, 40(1):599–656, January 2011.
5. Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. Linear temporal logic and propositional schemata, back and forth. In *Proceedings of the 2011 Eighteenth International Symposium on Temporal Representation and Reasoning*, TIME '11, pages 80–87, Washington, DC, USA, 2011. IEEE Computer Society.
6. Vincent Aravantinos, Mnacho Echenim, and Nicolas Peltier. A resolution calculus for first-order schemata. *Fundamenta Informaticae*, 2013.

[1] http://www.logic.at/staff/weller/index.html
[2] http://www.logic.at/people/leitsch/
[3] http://www.logic.at/staff/giselle/

7. Matthias Baaz. Note on the generalization of calculations. *Theoretical Computer Science*, 224(12):3 – 11, 1999.
8. Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Ceres: An analysis of Fürstenberg's proof of the infinity of primes. *Theor. Comput. Sci.*, 403(2-3):160–175, August 2008.
9. Matthias Baaz and Richard Zach. Short proofs of tautologies using the schema of equivalence. In Egon Brger, Yuri Gurevich, and Karl Meinke, editors, *Computer Science Logic*, volume 832 of *Lecture Notes in Computer Science*, pages 33–35. Springer Berlin Heidelberg, 1994.
10. E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer, 1997.
11. J.C. Bradfield. The modal mu-calculus alternation hierarchy is strict. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR '96: Concurrency Theory*, volume 1119 of *Lecture Notes in Computer Science*, pages 233–246. Springer Berlin Heidelberg, 1996.
12. Hubert Comon. Inductionless induction. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 913–962. Elsevier and MIT Press, 2001.
13. D. Cooper. Theorem proving in arithmetic without multiplication. In *Machine Intelligence*, 1972.
14. John Corcoran. Schemata: The concept of schema in the history of logic. *Bulletin of Symbolic Logic*, (2):219–240.
15. Cvetan Dunchev, Alexander Leitsch, Mikheil Rukhaia, and Daniel Weller. Ceres for first-order schemata. *CoRR*, abs/1303.4257, 2013.
16. Gerhard Gentzen. Fusion of several complete inductions. In M.E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, volume 55 of *Studies in Logic and the Foundations of Mathematics*, pages 309 – 311. Elsevier, 1969.
17. Jürgen Giesl and Deepak Kapur. Decidable classes of inductive theorems. In *Proceedings of the First International Joint Conference on Automated Reasoning*, IJCAR '01, pages 469–484, London, UK, UK, 2001. Springer-Verlag.
18. Deepak Kapur and Mahadavan Subramaniam. Extending decision procedures with induction schemes. In David McAllester, editor, *Automated Deduction - CADE-17*, volume 1831 of *Lecture Notes in Computer Science*, pages 324–345. Springer Berlin Heidelberg, 2000.
19. Jan Krajek and Pavel Pudlák. The number of proof lines and the size of proofs in first order logic. *Archive for Mathematical Logic*, 27(1):69–84, 1988.
20. V.P. Orevkov. Proof schemata in Hilbert-type axiomatic theories. *Journal of Soviet Mathematics*, 55(2):1610–1620, 1991.
21. R. J. Parikh. Some results on the length of proofs. *Transactions of the American Mathematical Society*, 177:pp. 29–36, 1973.
22. Gaisi Takeuti. *Proof Theory*, volume 81 of *Studies in logic and the foundations of mathematics*. American Elsevier Pub., 1975.