Tarskian Set Constraints

ROBERT GIVAN[1], **DEXTER KOZEN, DAVID MCALLESTER, and CARL WITTY**

Massachusetts Institute of Technology, Cambridge Massachusetts

Abstract: We investigate set constraints over set expressions with Tarskian functional and relational operations. Unlike the Herbrand constructor symbols used in recent set constraint formalisms, the meaning of a Tarskian function symbol is interpreted in an arbitrary first order structure. We show that satisfiability of Tarskian set constraints is decidable in nondeterministic doubly exponential time. We also consider various extensions of the basic language and show that: satisfiability of Tarskian set constraints with recursion (μ-sets) is undecidable but satisfiability for Tarskian set constraints with μ-sets but without function symbols is linear time equivalent to satisfiability in the propositional μ-calculus and is therefore decidable in deterministic exponential time.

Keywords: Set Constraints, Type Inference, Machine Inference, Theorem Proving, Automated Reasoning, Mechanical Verification, Modal logic, Concept Languages, Fixed Points, μ-calculus.

---

[1]This work supported by a Hertz graduate fellowship

1 Introduction

There has been considerable interest recently in formalisms for describing and reasoning about sets. Before describing Tarskian set constraints we give a brief overview of existing set calculi. We describe four general classes of set calculi.

First, and most recently, there are formalisms for "Herbrand" set constraints. A Herbrand set expression denotes a subset of the Herbrand universe over some given set of constructor function symbols. A Herbrand set expression is an expression which can be built recursively from Herbrand set variables, applications of constructors and projection functions, and Boolean combinations of Herbrand set expressions. For example, the expression $\mathtt{pair}(X, Y) \cup \mathtt{first}(Z)$ is the set of Herbrand terms which are either of the form $\mathtt{pair}(s,\ t)$ for $s \in X$ and $t \in Y$ or equal to t such that $\mathtt{pair}(t, w) \in Z$ for some term w. A Herbrand set constraint is an expression of the form $S \subseteq W$ or $S \not\subseteq W$ where S and W are Herbrand set expressions.

Herbrand set constraints have a wide variety of applications in program analysis and automated reasoning [Heintze and Jaffar, 1990], [Aiken and Wimmers, 1994]. Herbrand set constraints were introduced by Heintze and Jaffar who gave a procedure for solving systems of constraints of the form $S \subseteq W$ where W contains only set variables and function applications and S does not contain negation. Subsequently decision procedures have been given for ever increasing subclasses of Herbrand set constraints. Very recently it appears that a decision procedure has been given for satisfiability of an arbitrary system of Herbrand set constraints [**?**].

A second class of set formalisms have been developed as part of knowledge representation languages for artificial intelligence applications [Brachman and Schmolze, 1985], [Schmidt-Schaub and Smalka, 1991]. These languages provide a variety of primitives for constructing expressions denoting sets. They are often referred to collectively as "concept languages" or "terminological languages". The set expressions of concept languages are constructed from set variables and relation variables using a variety of compositional mechanisms. For example, the expression $\forall R.C$ where R is a relation expression and C is a set expression denotes the set $\{x :\ \forall y\ \ R(x, y) \rightarrow y \in C\}$. For the most part these languages can be viewed as fragments of propositional dynamic logic (PDL) discussed below [Calvanese *et al.*, 1994], [Giacomo and Lenzerini, 1994b], [Giacomo and Lenzerini, 1994a]. However, many of the languages correspond to highly restricted fragments of PDL. Many of them have satisfiability problems in P, NP, or PSPACE [Donini *et al.*, 1991]. This is significantly better than full PDL which is complete for exponential time. Also, concept languages often include cardinality primitives which appear not to be expressible in PDL. Furthermore, there is a natural relationship between certain concept languages and Montague grammar for natural language. In particular the set expression $R(\texttt{every}\ C)$ is taken to be the set $\{x:\ \forall y \in C\ \ R(x,y)\}$. This provides a natural meaning for English verb phrases such as "contains every prime number". One simple but expressive Montagovian concept language has a polynomial time satisfiability problem [McAllester and Givan, 1992].

A third class of set formalisms consists of modal and temporal logics. These logics involve "formulas" which are true or false of possible worlds in Kripke structures. However, it is natural to view a Kripke structure as an ordinary first order structure whose domain is the set of possible worlds. Under this view each formula in a modal or dynamic logic denotes a set of worlds, which is formally nothing more than a subset of the domain of the structure in which the formula is interpreted. In simple modal logics there is a distinguished binary relation R called the accessibility relation and the expression $\Diamond C$ denotes the set $\{x:\ \exists y\ R(x,y) \wedge y \in C\}$. Similarly $\Box C$ denotes $\{x:\ \forall y\ \ R(x,y) \rightarrow y \in C\}$. propositional dynamic logic (PDL) is a modal logic which involves relation expressions as well as set expressions [Pratt, 1976]. If R is a relation expression and C is a set expression then the PDL set expression $[R]C$ is equivalent to the concept expression $\forall R.C$ and denotes the set $\{x:\ \forall y\ \ R(x,y) \rightarrow y \in C\}$. The set expression $<\!R\!>C$ is defined analogously to denote $\{x:\ \exists y\ R(x,y) \wedge y \in C\}$. PDL also allows for a variety of relation expressions which include all regular expressions over a base set of relation constants. The propositional μ-calculus is a generalization of PDL which allows for recursively defined set expressions of the form $\mu X.C[X]$ where X is a set variable and $C[X]$ is a set expression in which every occurrence of X occurs inside an even number of negation signs [Kozen, 1983]. We will show below that the propositional μ-calculus is essentially equivalent to recursive Tarskian set constraints without function symbols.

The μ-calculus is discussed in greater detail in section 9. Satisfiability for the propositional μ-calculus is known to be complete for deterministic exponential time [Streett and Emerson, 1989], [Emerson and Jutla, 1988], [Safra, 1988].

The fourth, and oldest, class of set calculi are based on what we will call Tarskian set expressions. The simplest Tarskian set expressions are of the form $R(C_1, \ldots, C_n)$ where R is a relation of $n+1$ arguments. This expression denotes the set $\{y : \exists x_1 \in C_1, \ldots, x_n \in C_n : \quad R(x_1, \ldots, x_n, y)\}$. Tarskian set constraints were first studied extensively by Jónsson and Tarski who proved an analogue of the Stone representation theorem for an appropriate algebraic characterization of "Boolean algebra with operations" [Jònsson and Tarski, 1951] [Jònsson and Tarski, 1952].[2] Tarskian set constraints also include expressions of the form $f(C_1, \ldots, C_n)$ where f is a function symbol. This expression denotes the set $\{f(x_1, \ldots, x_n) : x_1 \in C_1, \ldots, x_n \in C_n\}$. A Tarskian set constraint is an expression of the form $S \subseteq W$ or $S \not\subseteq W$ where S and W are constructed from set variables, relation and function application as just described, and Boolean operations. Tarskian set expressions are interpreted relative to a first order structure. Syntactically, Tarskian set constraints are very similar to Herbrand set constraints. However, Herbrand set constraints, unlike Tarskian set constraints, are always interpreted over *the* Herbrand universe generated by a specified set of constructor functions. The fact that the function and relation symbols used in Tarskian set constraints can have arbitrary meanings greatly changes the nature of the satisfiability problem.

Tarskian set expressions can be extended in a natural way with least fixed point expressions of the form $\mu X C[X]$ where X is a set variable and $C[X]$ is a set expression that is syntactically monotone in X. Tarskian set expressions involving such μ-sets will be called *recursive*. It is shown here that the satisfiability problem for recursive Tarskian set constraints without function symbols is essentially equivalent to the satisfiability problem for the propositional μ-calculus. Hence recursive Tarskian set constraints in some sense subsume the propositional μ-calculus which in turn subsumes PDL

[2] Analogues of the Stone representation theorem for subclasses of Boolean algebras with operations have recently been studied in a general setting by Goldblatt [Goldblatt, 1989]. See also [Kozen, 1993].

which in turn subsumes most features of the concept languages developed for AI applications.

In spite of the apparent naturality of Tarskian set constraints their computational properties have not been widely studied. It is shown in [McAllester and Givan, 1993] that satisfiability of nonrecursive Tarskian set constraints not involving Boolean operations is decidable in cubic time (assuming unit time hash table operations). The results of this paper are summarized in the following table. We categorize Tarskian set constraint satisfiability problems by the presence or absence of recursion (μ-sets), the presence or absence of function symbols of arity greater than or equal to 1, and the presence or absence of constant symbols (function symbols of arity 0). In all cases we allow both positive and negative constraints.

	Recursion	Functions	Constants	Lower Bound	Upper Bound
1.	-	-	-	EXPTIME	EXPTIME
2.	-	-	+	EXPTIME	EXPTIME
3.	-	+	-	NEXPTIME	NEXPTIME
4.	-	+	+	NEXPTIME	2-NEXPTIME
5.	+	-	-	EXPTIME	EXPTIME
6.	+	-	+	EXPTIME	?
7.	+	+	-	NEXPTIME	?
8.	+	+	+	Undecidable	?

The results in the first two lines of the table are proved using techniques similar to those used for PDL. The lower bound in line three is proved using a reduction of a fragment of the monadic class. The upper bound in line three is proved by a filtration-like argument. There does not appear to be any simple reduction from Tarskian constraints to the monadic class.

Standard techniques fail for the fourth line, the case of nonrecursive constraints with arbitrary function symbols. We show that satisfiability for nonrecursive Tarskian set constraints is decidable in nondeterministic doubly exponential time. Our procedure involves a reduction to a natural class of diophantine constraints which we call *semi-linear*. We show that satisfiability for semi-linear diophantine constraints is decidable in nondeterministic exponential time. However, we conjecture that semi-linear diophantine satisfiability is in NP. If so, then we get a nondeterministic exponential procedure for nonrecursive Tarskian constraints.

The fifth line in the table corresponds to recursive constraints without function symbols of any kind. We show here that satisfiability for this Tarskian set constraint class is linear time equivalent (under Turing reductions) to satisfiability for the propositional μ-calculus — the two logics are essentially identical. The bounds in line 5 follow from known bounds for the propositional μ-calculus.

The eighth line shows that if we allow both recursion and arbitrary function symbols then the satisfiability problem becomes undecidable. The proof is by a reduction of Hilbert's tenth problem. It is interesting to note that the difficulties in both lines four and eight arise from the ability to express diophantine constraints. It seems that both constants and functions of arity at least one are necessary for expressing such constraints.

The sixth and seventh lines in the table are open. The reduction to the propositional μ-calculus appears to fail. However, the logics of lines six and seven do not seem sufficiently powerful to express diophantine constraints.

The following section gives a more precise definition of Tarskian set constraints. This is followed by sections roughly corresponding to the rows of the above table.

2 Tarskian Set Constraints

We assume a fixed but arbitrary signature consisting of a set of function and relation symbols each of which is associated with a specified arity (number of arguments). As usual, constants are treated as functions of no arguments. We also assume an countably infinite collection of set variables. Given such function symbols, relation symbols, and set variables, a *set expression* is defined recursively to be one of the following:

- a set variable
- a Boolean combination of the form $\neg C$, $(C_1 \cup C_2)$ or $(C_1 \cap C_2)$.

- an application $R(C_1, \cdots, C_n)$ where R is either an $n+1$ ary relation symbol or an n-ary function symbol.

- or a μ-expression of the form $\mu X.C[X]$ where X is a set variable and $C[X]$ is a set expression such that every occurrence of X in $C[X]$ occurs an even number of set complement operations.

The intuitive meaning of each type of set expression should be clear. Informally, the expression $R(C_1,\ \ldots,\ C_n)$ is the set $\{y : \exists x_1 \in C_1,\ \ \ldots,\ x_n \in C_n\ \ \langle x_1,\ \ldots,\ x_n,\ y\rangle \in R\}$. The set $\mu X.C[X]$ is the least fixed point of the operation from sets to sets which maps a set X to the set $C[X]$. For example $\mu X.(a \cup f(X))$ is the set of all domain elements of the form $f^n(a)$. A precise semantics is given below.

A model $\mathcal{M}$ for the given set of function and relation symbols consists of a domain plus an interpretation of each set variable as a subset of that domain, an interpretation of each n-ary relation as a set of n-tuples over D, i.e., a subset of D^n, and an interpretation of each n-ary function symbol as a functional input output relation which is a subset of D^{n+1}. More precisely if f is a function symbol of arity n then the meaning that $\mathcal{M}$ assigns to f, denoted, $\mathcal{M}[\![f]\!]$, must be a subset of D^{n+1} such that for every $\langle x_1,\ \ldots,\ x_n\rangle \in D^n$ there exists exactly one $y \in D$ such that $\langle x_1,\ \ldots,\ x_n,\ y\rangle \in \mathcal{M}(f)$. A model assigns a meaning to every Tarskian set expression in a natural way.

> **Definition:** Let $\mathcal{M}$ be a model with semantic domain D. For any set expression C we define the semantic interpretation of C, written $\mathcal{M}[\![C]\!]$, to be a subset of D determined by the following conditions.
>
> - For any set variable X we define $\mathcal{M}[\![X]\!]$ to be the set assigned to X by $\mathcal{M}$.
> - $\mathcal{M}[\![(C_1 \cup C_2)]\!]$ is the union of $\mathcal{M}[\![C_1]\!]$ and $\mathcal{M}[\![C_2]\!]$. $\mathcal{M}[\![(C_1 \cap C_2)]\!]$ is defined analogously. $\mathcal{M}[\![\neg C]\!]$ is the subset of D not contained in $\mathcal{M}[\![C]\!]$.
> - If R is an n-ary function symbol or $n+1$-ary relation symbol then $\mathcal{M}[\![R(C_1, \cdots, C_n)]\!]$ is the set of all y such that there

exist elements $x_1 \ldots, x_n$ in $\mathcal{M}[\![C_1]\!], \cdots, \mathcal{M}[\![C_n]\!]$ respectively such that $\langle x_1, \ldots, x_n, y\rangle \in \mathcal{M}(R)$.

- $\mathcal{M}[\![\mu X.C[X]]\!]$ is the intersection of all subsets U of D such that $U = \mathcal{M}[X := U][\![C[X]]\!]$ where $\mathcal{M}[X := U]$ is the model identical to $\mathcal{M}$ except that it interprets the set variable X as the set U.

A *Tarskian set constraint* is a formula of the form $C_1 \subseteq C_2$ or $C_1 \not\subseteq C_2$ where C_1 and C_2 are set expressions. The former are called positive constraints and the later are called negative constraints. If $\mathcal{M}$ is a model then we write $\mathcal{M} \models C_1 \subseteq C_2$ if $\mathcal{M}[\![C_1]\!]$ is a subset of of $\mathcal{M}[\![C_2]\!]$. We write $\mathcal{M} \models C_1 \not\subseteq C_2$ if $\mathcal{M}[\![C_1]\!]$ is not a subset of of $\mathcal{M}[\![C_2]\!]$. For any set Σ of Tarskian set constraints we write $\mathcal{M} \models \Sigma$ if $\mathcal{M} \models \Phi$ for each constraint Φ in Σ. We say that a set Σ of Tarskian set constraints constraint is *satisfiable* if there is a model $\mathcal{M}$ such that $\mathcal{M} \models \Sigma$.

3 Tarskian set constraints without Recursion or Functions

This section gives a simple (deterministic) exponential time procedure for determining the satisfiability of a system of nonrecursive Tarskian set constraints without function symbols (of any arity). Some terminology is also introduced which is used in later sections. The results in this section are subsumed by those of section 9 where we show that satisfiability for recursive Tarskian set constraints without functions is decidable in deterministic exponential time. However, the procedure in section 9 is based on the known decision procedure for the propositional μ-calculus which in turn is based on tree automaton techniques and is probably unusable in practice. Here we sketch a much simpler exponential time procedure similar to procedures used for PDL.

We start by defining some terminology that will be also be useful in later sections.

Definition: Let Σ be a set of nonrecursive Tarskian set constraints, i.e., set constraints not involving μ-sets. We say that a set expression C *occurs in* Σ if either C occurs as a top level set expression in a constraint in Σ or as a subexpression of such an expression. A Σ-type is a set τ of set expressions satisfying the following conditions:

- Every element of τ is of the form C or $setnotC$ where C occurs in Σ. and, for any C occuring in Σ, τ contains exactly one of C or $\neg C$.
- If $(C_1 \cup C_2)$ occurs in Σ then τ contains $(C_1 \cup C_2)$ if and only if τ contains at least one of C_1 and C_2.
- If $(C_1 \cap C_2)$ occurs in Σ then τ contains $(C_1 \cap C_2)$ if and only if τ contains both C_1 and C_2.
- If Σ contains $C \subseteq W$, and τ contains C, then τ contains W.

Intuitively, the Σ-types correspond to the "types" of domain elements where two elements have the same type if they can not be distinguished by set expressions occurring in Σ. The Σ-types are analogous to truth assignments on the Fischer-Ladner closure used in decision procedures for PDL [Fischer and Ladner, 1978]. Now consider a model $\mathcal{M}$.

Definition: A domain element x in a model $\mathcal{M}$ *inhabits* a Σ-type τ if for every $C \in \tau$ we have $x \in \mathcal{M}[\![C]\!]$. If some domain element inhabits τ then τ is called *inhabited.*

Next we define a notion of a "possible output" for an application of a relation symbol.

Definition: A type τ is a *possible output* of relation R applied to types $\sigma_1, \ldots, \sigma_n$, written $R(\sigma_1, \ldots, \sigma_n) \rightsquigarrow \tau$, provided that τ does not contain a set expression of the form $\neg R(C_1, \ldots, C_n)$ where $C_i \in \sigma_i$ for each C_i.

Intuitively, $R(\sigma_1, \ldots, \sigma_n) \rightsquigarrow \tau$ provided that the set expressions in τ do not forbid R from mapping objects of type σ_i to an object of type τ. The decision procedure given below is somewhat analogous to the exponential time decision procedure for PDL constructed by Pratt [Pratt, 1980].

> **Theorem:** Satisfiability of a system of nonrecursive Tarskian set constraints without function symbols is decidable in exponential time.

Proof: We define a simple procedure. Let Σ be a system of nonrecursive Tarskian set constraints. Initialize S to be the set of all Σ-types. Now repeatedly remove from S any type τ such that τ contains $R(C_1, \ldots, C_n)$ but there are no types $\sigma_1, \ldots, \sigma_n$ in S such that each σ_i contains C_i and $R(\sigma_1, \ldots, \sigma_n) \rightsquigarrow \tau$.

The final set S of Σ-types can be computed in time that is exponential in the number of set expressions occurring in Σ. We now show that Σ is satisfiable if and only if for all negative constraints $C \not\subseteq W$ in Σ there exists some type τ in S such that τ contains C but not W. First suppose there is some negative constraint $C \not\subseteq W$ in Σ such that every type in S that contains C also contains W. Let $\mathcal{M}$ be an arbitrary model satisfying all positive constraints $U \subseteq V$ in Σ. We will show that $\mathcal{M}$ must violate $C \not\subseteq W$ and hence Σ is unsatisfiable. For any element x of the domain of Σ let the type of x be the unique Σ-type τ such that for all set expressions U occurring in Σ, τ contains U if and only if $x \in \mathcal{M}[\![U]\!]$. It is not difficult to show that every Σ-type removed by the procedure must be uninhabited in $\mathcal{M}$. Therefore the type of any element of the domain of $\mathcal{M}$ must be an element of S. Now suppose that there is some element x of the domain of $\mathcal{M}$ such that $x \in \mathcal{M}[\![C]\!]$ but $x \notin \mathcal{M}[\![W]\!]$. In this case the type of x must contain C but not W. Since we have assumed that no such type is contained in S we have a contradiction and so no such x can exist. Hence $\mathcal{M} \models C \subseteq W$ so Σ is unsatisfiable.

Now suppose that for each negative constraint $C \not\subseteq W$ in Σ there exists a type in S containing C but not W. Let $\mathcal{M}$ be the model whose domain is the set S and such that $\mathcal{M}(X)$ is the set of types in S containing X and $\mathcal{M}(R)$ is the relation containing those tuples $\langle\sigma_1, \ldots, \sigma_n, \tau\rangle$ such that

$R(\sigma_1, \ldots, \sigma_n) \rightsquigarrow \tau$. One can prove, by structural induction on set expressions, that for any set expression C occurring in Σ we have that $\mathcal{M}[\![C]\!]$ is exactly the set of types in S containing C. It is not difficult to show that this implies that $\mathcal{M} \models \Sigma$. The case of set variables is true by definition. The case of union and intersection set expressions that occur in Σ is straightforward. If $\neg C$ occurs in σ then by the induction hypothesis $\mathcal{M}[\![C]\!]$ is the set of types in S containing C. Since every type contains exactly one of C and $\neg C$, we have that $\mathcal{M}[\![\neg C]\!]$, which equals $S - \mathcal{M}[\![C]\!]$, is precisely those types in S which contain $\neg C$. Now consider an application expression $R(C_1, \ldots, C_n)$. Let τ be a type in S containing this application expression. Since τ was not removed in the process of constructing S there must exist types $\sigma_1, \ldots, \sigma_n$ such that $R(\sigma_1, \ldots, \sigma_n) \rightsquigarrow \tau$ and where $C_i \in \sigma_i$ for each C_i. By the induction hypothesis $\sigma_i \in \mathcal{M}[\![C_i]\!]$ It follows from the definition of $\mathcal{M}(R)$ that $\tau \in \mathcal{M}[\![R(C_1, \ldots, C_n)]\!]$. Finally suppose that τ is a type in $\mathcal{M}[\![R(C_1, \ldots, C_n)]\!]$. By the definition of $\mathcal{M}(R)$ there must exist types $\sigma_1, \ldots, \sigma_n$ in $\mathcal{M}[\![C_1]\!], \ldots, \mathcal{M}[\![C_n]\!]$ respectively such that $R(\sigma_1, \ldots, \sigma_n) \rightsquigarrow \tau$. By the induction hypothesis we must have $C_i \in \sigma_i$ for each C_i. Now by the definition of $\rightsquigarrow$ we must also have that τ does not contain $\neg R(C_1, \ldots, C_n)$. But for every set expression occurring in Σ, τ contains either that expression or its negation. Hence τ must contain $R(C_1, \ldots, C_n)$.■

The proof of the above theorem yields a finite model property for Tarskian set constraints without recursion or function symbols. It is also possible, although we will not do it here, to give a simple set of inference rules and show that the steps of the above procedure can be simulated by inferences in that system and hence that the rules are sound and complete for this case.

4 The Lower Bound for the Easiest Case

In this section we show that satisfiability of Tarskian set constraints is EXPTIME hard for constraints without recursion or functions. The results of this section can be contrasted with known results on the satisfiability of individual set expressions in these expressively weak languages [Donini *et al.*, 1991]. In the nonrecursive case satisfiability of individual set expressions is considerably easier than satisfiability of a system of set constraints. We show in section 9 that when recursive set expressions are allowed, but functions are not, then constraint set satisfiability can be reduced to set expression satisfiability. Without recursion the reduction fails.

It turns out that languages somewhat weaker than Tarskian set constraints without recursion or functions are still hard for exponential time. We will characterize some weaker languages using the following definitions.

> **Definition:** If Σ is a set of constraints and Φ is a constraint we write $\Sigma \models \Phi$ to indicate that any model satisfying all the constraints in Σ also satisfies Φ. A *positive entailment problem* is a set Σ of *positive* set constraints and a positive set constraint Φ. The problem is to determine whether $\Sigma \models \Phi$.

A positive entailment problem is equivalent to set constraint satisfiability problem in which there is exactly one negative constraint. Now consider the following ways in which set expressions can be formed.

$\cup$ If U and W are set expressions then so is $U \cup V$.

$\cap$ If U and W are set expressions then so is $U \cap V$.

$\Diamond$ If C is a set expression and R is a binary relation symbol then $R(C)$, which will also be written as $<R>\, C$, is also a set expression.

$\Box$ If C is a set expression and R is a binary relation symbol then $\neg R(\neg C)$, which will also be written as $[R]C$, is also a set expression.

I will use the notation $\mathcal{L}(\mathcal{F}_1,\ \ldots,\ \mathcal{F}_n)$ to mean the set language with set variables and set formation features $\mathcal{F}_1, \ldots, \mathcal{F}_n$. For example, $\mathcal{L}(\cup,\ \Diamond)$ is the languages whose set expressions are constructed from set variables using only the set formation operations $\cup$ and $\Diamond$ as defined above. All of the languages defined by the above four features are sublanguages of nonrecursive Tarskian set constraints without function symbols. In all of these languages the only occurrences of the set complement operation are the occurrences implicit in set expressions of the form $[R]C$. We show here that the positive entailment satisfiability problem for $\mathcal{L}(\Box, \Diamond, \cap)$ and $\mathcal{L}(\Box, \Diamond, \cup)$ are both EXPTIME hard. Before doing this we briefly mention the difficulty of the positive entailment problem for other combinations of these features. The positive entailment problem for $\mathcal{L}(\cup)$ and $\mathcal{L}(\cap)$ can both be reduced in linear time to the satisfiability problem for propositional Horn clauses which is known to be decidable in linear time [Downing and Gallier, 1984]. The positive entailment problem for $\mathcal{L}(\cup, \cap)$ can be shown to be NP complete (we leave this as an exercise for the reader). It is known that satisfiability of Tarskian set constraints not involving Boolean operations or recursion, but with function symbols, is decidable in polynomial time [McAllester and Givan, 1993]. This implies that the positive entailment problem for $\mathcal{L}(\Diamond)$ is decidable in polynomial time. By duality arguments given below this implies that the positive entailment problem for $\mathcal{L}(\Box)$ is also decidable in polynomial time. To our knowledge the difficulty of the positive entailment problem for other combinations of these features is open.

It is possible to relax the semantics of Tarskian set expressions so that the "set" expressions denote elements of a lattice and $\cup$ and $\cap$ denote least upper bound and greatest lower bound operations respectively. Once we allow an arbitrary lattice (rather than require a complemented distributive lattice), and only require that relations denote monotone operations on lattice elements, then the positive entailment problems for $\mathcal{L}(\Diamond, \cup, \cap)$ is decidable in polynomial time [Givan and McAllester, 1992].

> **Theorem:** The positive entailment problem for $\mathcal{L}(\Box, \Diamond, \cap)$ is EXPTIME hard.

Proof: The proof is by reduction of the acceptance problem for linear space bounded alternating Turing machines. In an alternating Turing machine the states are classified into "universal" and "existential" states and for any given state and input symbol there can be many different next states (as in simple nondeterministic machines). A configuration of the machine consists of a state of the tape, the tape location of the Turing machine head, and the state of the machine. A configuration in which the machine is in a universal (existential, accepting) state is called a universal (existential, directly accepting) configuration. Each configuration has a set of possible next configurations defined by the transition table of the machine in the standard way. A configuration is called accepting if either it is directly accepting (the machine is in an accept state) or it is a universal configuration and all next configurations are accepting, or it is an existential configuration and some next configuration is accepting. The linear space alternating Turing machine problem can be phrased as the problem of deciding if a given configuration of a given alternating machine is accepting subject to the restriction that configurations are restricted to ones in which the head occurs on a given set of tape squares (all other configurations are taken to be failing configurations). We can assume without loss of generality that all configurations have exactly two next configurations. We show that this problem is polynomial time reducible to the positive entailment problem for $\mathcal{L}(\Box, \Diamond, \cap)$.

In the reduction to from alternating machines to set expressions the set expressions should be viewed as sets of machine configurations. Actually, it seems clearer to think of each set expression as a "proposition" which is true or false of any given configuration. Given any linear space bounded alternating Turing machine we introduce a set variable $X_{n,a}$ for each tape location n and possible tape symbol a. Intuitively $X_{n,a}$ represents the proposition that symbol a is written on square n. We also introduce a set variable H_n for each tape location n representing the proposition that the head is at square n and a proposition Z_s for each machine state s representing the proposition that the machine is in state s. We also have set variables `START` and `ACCEPT` representing the proposition that the current configuration is the given initial configuration and that the current configuration is an accepting configuration respectively. Finally we have one binary relation symbol N representing the "next configuration" relation. We let Σ be the following set of positive constraints. Each constraint should be viewed as an implication required to hold at all configurations.

$\mathtt{START} \subseteq H_0$

$\mathtt{START} \subseteq Z_{s_0}$ where s_0 is the initial state.

$\mathtt{START} \subseteq X_{n,a}$ where a is the initial symbol on tape square n.

A constraint of the form $X_{n,a} \cap H_n \cap Z_s \subseteq <N> (H_{n+1} \cap Z_w \cap X_{n,b})$ for each entry in the transition table of the machine which replaces a by b, moves from state s to state w, and moves right. A similar constraint is included for each left-moving entry in the machine table.

All constraints of the form $X_{n,a} \cap H_m \subseteq [N]X_{n,a}$ where n is not equal to m.

All constraints of the form $Z_s \subseteq \texttt{Accept}$ where s is an accepting state.

All constraints of the form $Z_s \cap <N> \texttt{Accept} \subseteq \texttt{Accept}$ where s is an existential state.

All constraints of the form $Z_s \cap <N> (Z_u \cap \texttt{Accept}) \cap <N> (Z_w \cap \texttt{Accept}) \subseteq \texttt{Accept}$ where s is a universal state with successor states u and w.

We leave it as an exercise for the reader to verify that $\Sigma \models \texttt{START} \subseteq \texttt{ACCEPT}$ if and only if the given initial configuration is accepting.■

We now establish a general duality principle for all languages defined by subsets of the language features discussed in this section.

Definition: If C is a concept expression then we define the dual of C, denoted as C^*, to be the result of simultaneously replacing $\Diamond$ by $\Box$, $\Box$ by $\Diamond$, $\cup$ by $\cap$, and $\cap$ by $\cup$.

Definition: For any model $\mathcal{M}$ we define $\mathcal{M}^*$ to be the interpretation which is identical to $\mathcal{M}$ except that for each set variable X, $\mathcal{M}^*[\![X]\!]$ is $\mathcal{M}[\![\neg X]\!]$.

Lemma: For any class expression C and model $\mathcal{M}$ we have that $\mathcal{M}^*[\![C^*]\!]$ equals $\mathcal{M}[\![\neg C]\!]$.

Proof: Push the negation in $\neg C$ down to the set variables using de Morgan's laws and the identities $\neg <R> C = [R]\neg C$ and $\neg[R]C = <R> \neg C$

Definition: For any constraint $C \subseteq W$, we define the dual constraint $(C \subseteq W)^*$ to be $W^* \subseteq C^*$ and for any set Σ of constraints we define Σ^* to be $\{C^* : \ C \in \Sigma\}$.

The preceding lemma implies that for any constraint Φ we have that $\mathcal{M} \models \Phi$ if and only if $\mathcal{M}^* \models \Phi^*$. This yields the following duality lemma.

> **Duality Lemma:** The entailment relation $\Sigma \models \Phi$ holds if and only if the dual relation $\Sigma^* \models \Phi^*$ holds.

This last lemma allows a direct reduction of the positive entailment problem for $\mathcal{L}(\Diamond, \Box, \cap)$ to the corresponding problem for $\mathcal{L}(\Diamond, \Box, \cup)$ so the latter must also be EXPTIME hard.

> **Corollary:** The positive entailment problem for $\mathcal{L}(\Box, \Diamond, \cup)$ is EXPTIME hard.

5 Nonrecursive Constraints without Functions of Arity Greater than 0

In this section we sketch a proof that satisfiability for Tarskian set constraints without recursion (μ-sets) and without functions of arity greater than zero is decidable in exponential time. The decision procedure is very similar to the procedure of the preceding section except that rather than deterministically construct a set of Σ-types we guess a set of types.

> **Theorem:** Satisfiability for nonrecursive Tarskian set constraints not involving μ-terms or function symbols of arity greater than 0 is decidable in exponential time.

Proof: We give a nondeterministic exponential time procedure involving only polynomially many nondeterministic choices. Any such procedure can be run in deterministic exponential time. First we guess a Σ-type for each constant symbol that appears in Σ. Note that this can be done with quadratically many nondeterministic choices (for each constant in Σ, and each set expression in Σ, we need to decide whether that set expression is in the Σ-type associated with the constant). Now initialize S to be all Σ-types not containing any constants plus the selected Σ-types for the constants. Then, as in the previous section, repeatedly remove from S any type τ such that τ contains $R(C_1, \ldots, C_n)$ but there are no types $\sigma_1, \ldots, \sigma_n$ in S such that each σ_i contains C_i and $R(\sigma_1, \ldots, \sigma_n) \rightsquigarrow \tau$. Accept Σ as satisfiable if the removal process does not remove any of the types containing constants and for each negative constraint $U \not\subseteq W$ in Σ there exists a type in the final set S that contains U but not W.

The proof of correctness of this procedure is a straightforward modification of the proof of the preceding section. Every type eliminated by the procedure must be uninhabited in any model corresponding to the choice of types for constants. Therefore, if the procedure does not accept Σ then Σ must be unsatisfiable. On the other hand if the procedure does accept Σ then the final set S of types constructed by the procedure can serve as the domain of a model of S.■

6 Nonrecursive Constraints without Functions of Arity 0

In this section we show that satisfiability for Tarskian set constraints without functions of arity 0 can be decided in nondeterministic exponential time. We start by proving the following lemma which states that when no constant symbols are involved it suffices to consider nonstandard models in which function symbols are allowed to denote "total" relations.

Definition: Let S be a subset of D^{n+1}. The relation S is called *total* if for every tuple $\langle x_1, \ldots, x_n \rangle \in D^n$ there exists some $y \in D$ such that $\langle x_1, \ldots, x_n, y \rangle \in S$. A *nonstandard model* is a model $\mathcal{M}$ such that for each function symbol f we have that $\mathcal{M}(f)$ is a total (but not necessarily functional) relation.

The meaning of a set expression in a nonstandard model is defined in an identical manner to its meaning in standard models. We can now state the following lemma for Tarskian set constraints without constant symbols. Note that the following lemma applies to recursive as well as nonrecursive constraints.

> **Lemma:** If Σ is a set of Tarskian set constraints not involving constant symbols then Σ is satisfiable if and only if Σ is satisfied by a nonstandard model.

Proof: Since every model is a nonstandard model the only if direction is trivial. Now suppose that there is a nonstandard model $\mathcal{M}$ which satisfies Σ. Let D be the domain of $\mathcal{M}$. By the Löwehheim-Skolem theorem for first order logic we can assume without loss of generality that D is countable. We will construct a standard model satisfying Σ whose domain is $D \times \omega$, i.e., the set of pairs $\langle x, i\rangle$ for $x \in D$ and i a natural number. We define a *target constraint* to be a tuple of the form $\langle f,\ x_1,\ \ldots,\ x_n,\ \langle y, k\rangle\rangle$ where f is a function symbol occurring in Σ, each x_i and y is in D, k is a natural number, and $\langle x_1,\ \ldots,\ x_n,\ y\rangle \in \mathcal{M}(f)$. Intuitively, a target constraint states that f should map the tuple $\langle x_1,\ \ldots,\ x_n\rangle$ to the pair $\langle y, k\rangle$. We define a *domain constraint* to be a tuple of the form $\langle f,\ \langle x_1, j_1\rangle,\ \ldots,\ \langle x_n, j_n\rangle\rangle$. Intuitively a domain constraint expresses the constraint that f must be defined on the tuple $\langle\langle x_1, j_1\rangle,\ \ldots,\ \langle x_n, j_n\rangle\rangle$. The set of all target and domain constraints is itself countable. Let $E_1,\ E_2,\ E_3,\ \ldots$ be an enumeration of all target and domain constraints (each target and domain constraint appears somewhere in this enumeration). We now define a series of models $\mathcal{M}_1,\ \mathcal{M}_2,\ \mathcal{M}_3,\ \ldots$ all of which have domain $D \times \omega$. The model $\mathcal{M}_1$ is defined by the following conditions:

- For any set variable X, $\mathcal{M}_1(X)$ is the set of pairs of the form $\langle x, i\rangle$ where $x \in \mathcal{M}(X)$.
- For any relation symbol R occurring in Σ, $\langle\langle x_1, j_1\rangle,\ \ldots,\ \langle x_n, j_n\rangle,\ \langle y, k\rangle\rangle \in \mathcal{M}_1(R)$ if and only if $\langle x_1,\ \ldots,\ x_n,\ y\rangle \in \mathcal{M}(R)$.
- For any function symbol f occurring in Σ, $\mathcal{M}_1(f)$ is empty.

The model $\mathcal{M}_{i+1}$ is defined in terms of $\mathcal{M}_i$ and the constraint E_i. If E_i is a target constraint $\langle f,\ x_1,\ \ldots,\ x_n,\ \langle y, k\rangle\rangle$ then we let $\mathcal{M}_{i+1}$ be identical to $\mathcal{M}_i$ except that $\mathcal{M}_{i+1}(f)$ contains one more tuple than $\mathcal{M}_i(f)$ and that tuple is of the form $\langle\langle x_1, j_1\rangle,\ \ldots,\ \langle x_n, j_n\rangle,\ \langle y, k\rangle\rangle$ where $\mathcal{M}_i(f)$ does not contain any tuple of the form $\langle\langle x_1, j_1\rangle,\ \ldots,\ \langle x_n, j_n\rangle,\ z\rangle$. If E_i is a domain expression $\langle f,\ \langle x_1, j_1\rangle,\ \ldots,\ \langle x_n, j_n\rangle\rangle$ then if $\mathcal{M}_i(f)$ contains a tuple of the form $\langle\langle x_1, j_1\rangle,\ \ldots,\ \langle x_n, j_n\rangle,\ z\rangle$ then $\mathcal{M}_{i+1}$ equals $\mathcal{M}_i$, otherwise $\mathcal{M}_{i+1}(f)$ contains one more tuple than $\mathcal{M}_i(f)$ and that tuple is of the form $\langle\langle x_1, j_1\rangle,\ \ldots,\ \langle x_n, j_n\rangle,\ \langle y, 0\rangle\rangle$ where $\langle x_1,\ \ldots,\ x_n,\ y\rangle \in \mathcal{M}(f)$. Such a y must exist because $\mathcal{M}(f)$ is total.

We let $\mathcal{M}'$ be the model such that $\mathcal{M}'(R)$ is the union of all relations of the form $\mathcal{M}_i(R)$ for any function or relation symbol R. It is not difficult to verify that for any function symbol f, $\mathcal{M}'(f)$ is functional. Hence $\mathcal{M}'$ is a standard model. Furthermore, one can also verify by structural induction on set expressions that for any set expression C not involving constants we have that $\mathcal{M}'[\![C]\!]$ is the set of pairs $\langle x, i\rangle$ such that $x \in \mathcal{M}[\![C]\!]$. This implies that $\mathcal{M}' \models \Sigma$.■

We now show that a set Σ of nonrecursive Tarskian set constraints without constants is satisfiable if and only if there exists a set of Σ-types satisfying certain easily checked conditions. This gives a simple nondeterministic exponential time procedure which simply guesses an appropriate set of Σ-types.

Definition: A set S of Σ-types is *locally consistent* if it satisfies the following conditions:

- For each negative constraint $U \not\subseteq W$ in Σ there is a type in S which contains U but not W.
- If a type τ in S contains a set expression of the form $R(C_1,\ \ldots,\ C_n)$ then there exist types $\sigma_1, \ldots, \sigma_n$ in S such that $R(\sigma_1,\ \ldots,\ \sigma_n) \rightsquigarrow \tau$ and $C_i \in \sigma_i$ for each C_i.
- For each function symbol f and Σ-types $\sigma_1,\ \ldots,\ \sigma_n$ in S, where n is the arity of f, there exists a Σ-type τ in S such that $f(\sigma_1,\ \ldots,\ \sigma_n) \rightsquigarrow \tau$.

Lemma: Let Σ be a set of nonrecursive Tarskian set constraints not involving functions of arity 0. Σ is satisfiable if and only if there exists a locally consistent set of Σ-types.

Proof Sketch: The proof is very similar to the proofs in the preceding two sections. If Σ is satisfiable then let $\mathcal{M}$ be a model of Σ and take S to be the set of types τ which are inhabited in $\mathcal{M}$. It is not difficult to show that S satisfies the above conditions. Conversely, let S be a set of types satisfying the above conditions. Let $\mathcal{M}$ be the structure whose domain is the set S of types and where the set variables and function and relation symbols are interpreted as follows. As in the earlier proofs interpret each set variable and monadic relation symbol as the set of types in S that contain that symbol. We interpret each function or relation symbol R so that $\mathcal{M}(R)$ is the relation containing those tuples $\langle \sigma_1, \ldots, \sigma_n, \tau \rangle$ such that $R(\sigma_1, \ldots, \sigma_n) \rightsquigarrow \tau$. The argument given in section 3 can be used to show that for each type $\tau \in S$ and each set expression C occurring in Σ we have that $\tau \in \mathcal{M}[\![C]\!]$ if and only if $C \in \tau$. This can be used to show that $\mathcal{M}$ is a model of Σ.

Although this procedure is quite similar to the procedures of sections 3 and 4, it requires a potentially exponential degree of nondeterminism. We can attempt to construct the set S of types by initializing it to be all types an then eliminating those types which are provably empty in any model. A problem arises however when attempting to satisfy the final condition on the set S. Suppose that there is a function symbol f and Σ-types $\sigma_1, \ldots,$ σ_n in S, where n is the arity of f, such that there is no $\tau \in S$ such that $f(\sigma_1, \ldots, \sigma_n) \rightsquigarrow \tau$. In this case one of the σ_i must be removed from S but we are left with a choice of which σ_i to remove.

7 Lower Bound for Nonrecursive Constraints Without Constants

In this section we give a reduction from a fragment of the monadic class known to be complete for nondeterministic exponential time to nonrecursive Tarskian constraints with functions but without constants.

> **Definition:** A *first order clause* is a first order sentence of the form $\forall x_1, \ldots x_n \Psi_1 \vee \ldots \vee \Psi_k$ where each Ψ_i is a first order literal, i.e., either an application of a predicate symbol to terms or the negation of such an application. Let a be a fixed constant symbol and let f be a fixed monadic function symbol. We define a *Lewis clause* (over a and f) to be one of the following:
>
> - An atomic sentence of the form $P(a)$.
> - A clause involving a single variable x where every literal contains an application of a monadic predicate to either x or $f(x)$.
> - A clause involving exactly two variables in which every literal contains an application of a monadic predicate to one of the two variables.

The following result is due to Lewis [Lewis, 1980].

> **Theorem (Lewis):** Satisfiability for a set of Lewis clauses is complete for NEXPTIME.

Note that Lewis clauses involve only monadic predicates. Function symbols only arise in clauses of the second type. It is not difficult to show that a set of Lewis clauses can be "inverse Skolemized" to produce an equisatisfiable sentence without any function symbols and involving only monadic predicates. Hence Lewis clauses can be viewed as a fragment of the monadic class. The NEXPTIME lower bound for Herbrand set constraints established in [Bachmair *et al.*, 1993] was also proved using a reduction of Lewis clauses.

Theorem: Satisfiability for nonrecursive Tarskian set constraints without constants is NEXPTIME hard.

Proof: The proof is by reduction of Lewis clauses. Let $\mathcal{C}$ be a set of Lewis clauses. We define a set $T[\mathcal{C}]$ of nonrecursive Tarskian set constraints as follows. Let $P_1(a)$, ... $P_n(a)$ be all clauses in $\mathcal{C}$ of the form $P(a)$. We include in $T(\mathcal{C})$ the constraint $P_1 \cap \ldots \cap P_n \not\subseteq \mathbf{F}$ where $\mathbf{F}$ is the empty set expression $X \cap \neg X$ for an arbitrary set variable X. For each clause of the form $P_1(x) \vee \ldots \vee P_n(x) \vee Q_1(f(x)) \vee \ldots \vee Q_m(f(x))$ in $\mathcal{C}$, where each P_i and Q_i is either a predicate or its negation, we include the constraint

$$f(\neg P_1 \cap \ldots \cap \neg P_n) \subseteq Q_1 \cup \ldots \cup Q_m$$

in $T[\mathcal{C}]$. Finally, consider a clause n $\mathcal{C}$ of the form $P_1(x) \vee \ldots \vee P_n(x) \vee Q_1(y) \vee \ldots \vee Q_m(y)$ where each P_i and Q_i is either a predicate or its negation. Let g be a fixed but arbitrary binary function symbol. For each such clause in $\mathcal{C}$ we include the constraint

$$g(\neg P_1 \cap \ldots \cap \neg P_n, \neg Q_1 \cap \ldots \cap \neg Q_m) \subseteq \mathbf{F}$$

in $T[\mathcal{C}]$.

It is easy to show $T[\mathcal{C}]$ and $\mathcal{C}$ are equisatisfiable. Suppose that $\mathcal{C}$ is satisfied by model $\mathcal{M}$. If we extend $\mathcal{M}$ by interpreting g as the constant function which maps all pairs of domain elements to the value of a then we get a model of $T[\mathcal{C}]$. Conversely, let $\mathcal{M}$ be a model of $T[\mathcal{C}]$. It must be possible to extend $\mathcal{M}$ by interpreting a in such a way as to satisfy all clauses of the form $P(a)$ in $\mathcal{C}$. This extension is a model of $\mathcal{C}$.■

8 Nonrecursive Tarskian Constraints

We now consider the problem of determining satisfiability for nonrecursive Tarskian set constraints. The procedure presented in this section nondeterministically reduces set constraint satisfiability to a system of diophantine constraints. The first step of the procedure guesses a locally consistent set of inhabited Σ-types as defined in section 6. The procedure then constructs a system of diophantine inequalities describing cardinality constraints. In order to describe the cardinality constraints we need some additional terminology.

Consider an element x which inhabits a Σ-type σ in a model $\mathcal{M}$. For each such application expression $R(C_1, \ldots, C_n)$ in σ there must exist elements $y_1, \ldots, y_n$ in $\mathcal{M}[\![C_1]\!], \ldots, \mathcal{M}[\![C_n]\!]$ respectively such that $\langle y_1, \ldots, y_n, x\rangle \in \mathcal{M}(R)$ and such that $y_i \in \mathcal{M}[\![C_i]\!]$. The values $y_1, \ldots, y_n$ can be viewed as "predecessors" of x which "justify" the fact that x is in the set $R(C_1, , \ldots, C_n)$. The predecessors $y_1, \ldots, y_n$ each have Σ-types $\tau_1, \ldots, \tau_n$. These observations lead to the following definitions.

Definition: Let S be a set of Σ-types. A *range expression* is an expression of the form $f(\sigma_1, \ldots, \sigma_n)$, where $\sigma_1, \ldots, \sigma_n$ are Σ-types in S and f is a function symbol. We say that a domain element d of a model $\mathcal{M}$ *inhabits* a range expression $f(\sigma_1, \ldots, \sigma_n)$ if there are some domain elements $d_1, \ldots, d_n$ inhabiting Σ-types $\sigma_1, \ldots, \sigma_n$ respectively such that $\langle d_1, \ldots, d_n, d\rangle \in \mathcal{M}(f)$. We will say that a range expression is *inhabited* in $\mathcal{M}$ when some element of the domain of $\mathcal{M}$ inhabits that expression.

Definition: A *predecessor justification* for a set $f(C_1, \ldots, C_n)$ is a range expression $f(\tau_1, \ldots, \tau_n)$ such that $C_i \in \tau_i$. A Σ-*predecessor-type* is a pair $\langle\sigma, \Delta\rangle$ where σ is a Σ-type and Δ is a mapping from function applications appearing in the type σ to range expressions such that for any function application U in σ we have that $\Delta(U)$ is a predecessor justification of U and $\Delta(U) \leadsto \sigma$. An object x inhabits a Σ-predecessor-type $\langle\sigma, \Delta\rangle$ in a model $\mathcal{M}$ if x inhabits σ and, for each application expression U in σ, x inhabits $\Delta(U)$.

An application expression $f(C_1, \ldots, C_n)$ can often be justified in more than one way, i.e., many different predecessors of many different Σ-types can cause x to be in the set $f(C_1, \ldots, C_n)$. However, for each element x in a model of Σ there will be at least one Σ-predecessor-type $\langle\sigma, \Delta\rangle$ inhabited by x. We will assume that some choice function is provided so that for each element x of the domain of $\mathcal{M}$ we can choose a unique a Σ-predecessor-type for x.

Note that the number of Σ-types is at most $2^{|\Sigma|}$. The number of justification functions Δ is no greater than the number of functions from expressions in Σ to range expressions. We can assume without loss of generality that no application expressions involve more than two arguments (larger arities can be eliminated with the use of pairing functions). Under this assumption there are at most $|\Sigma|2^{2|\Sigma|}$ range expressions. Hence the number of justification functions Δ is no more than $(|\Sigma|2^{2|\Sigma|})^{|\Sigma|}$ which is $2^{2|\Sigma|^2+|\Sigma|\log|\Sigma|}$ and hence is exponential in Σ. This implies that the number of Σ-predecessor-types is also exponential in $|\Sigma|$.

> **Definition:** Let S be a set of Σ-types. For each type $\sigma \in S$ let z_σ be a variable representing the number of inhabitants of σ. For each range expression $f(\tau_1, \ldots, \tau_n)$ with $\tau_i \in S$ let $z_{f(\tau_1, \ldots, \tau_n)}$ be a variable representing the number of domain members x such that $f(\tau_1, \ldots, \tau_n)$ is a justification in the selected predecessor type of x. For each Σ-predecessor-type $\langle\sigma, \Delta\rangle$ let $z_{\langle\sigma, \Delta\rangle}$ be a variable representing the number of individuals whose selected predecessor type is $\langle\sigma, \Delta\rangle$. We define $D(S)$ to be the semi-linear system of diophantine constraints including the following constraints:
>
> 1. **(Habitation Constraint)** For every type $\sigma \in S$, z_σ must be greater than one.
> 2. **(Range constraint)** $z_{f(\tau_1, \ldots, \tau_n)}$ must equal the sum of all $z_{\langle\sigma, \Delta\rangle}$ such that $\Delta(U) = f(\tau_1, \ldots, \tau_n)$ for some function application $U \in \sigma$.
> 3. **(Type constraint)** z_σ must equal the sum of all $z_{\langle\sigma, \Delta\rangle}$.
> 4. **(Predecessor Constraint)**. $z_{f(\tau_1, \ldots, \tau_n)}$ must not exceed the product of the z_{τ_i}. If f is a constant symbol, and hence there are no τ_i, then z_f must equal 1.

We now come to the first main theorem of this section.

Theorem: A set Σ of nonrecursive Tarskian constraints is satisfiable if and only if there exists a locally consistent set S of Σ-types such that the constraint set $D(S)$ is satisfiable over the positive integers plus ∞.

Proof: First suppose that Σ is satisfied by a model $\mathcal{M}$. Let S be the set of Σ-types inhabited by $\mathcal{M}$. It is easy to check that S is locally consistent. Now select for each element x in the domain of $\mathcal{M}$ a Σ-predecessor-type which it inhabits. We interpret the cardinality variable $z_{\langle\sigma,\ \Delta\rangle}$ to be the cardinality of the set of elements whose selected Σ-predecessor-type is $\langle\sigma,\ \Delta\rangle$. If this cardinality is infinite then we assign $z_{\langle\sigma,\ \Delta\rangle}$ the special value ∞ and ignore the order of infinity of the actual cardinality. The constraints in $D(S)$ then determine the interpretation of the variables z_σ and $z_{f(\tau_1,\ \ldots,\ \tau_n)}$. It is not difficult to check that all constraints in $D(S)$ are satisfied under this interpretation.

Now suppose that there exists a locally consistent set S of Σ-types such that $D(S)$ is satisfiable over positive integers plus ∞. We will build a model of Σ. The domain of our model will be the union over all inhabited types σ of sets $\{\,\sigma^i \mid i \leq z_\sigma\,\}$ where for each type σ and each index i we assume that σ^i is a distinct object. If z_σ is ∞ then there will be a countably infinite number of distinct elements of the form σ^i.

We define $\mathcal{M}$ on each set variable X to be the set of all domain elements σ^i whose base type σ contains X. We define $\mathcal{M}(R)$ for a relation symbol R be the set of all tuples $\langle(\sigma_1)^{(i_1)}, \ldots, (\sigma_n)^{(i_n)},\ \tau^j\rangle$ such that $R(\sigma_1,\ \ldots,\ \sigma_n) \leadsto \tau$.

Finally, we define $\mathcal{M}$ on the function symbols. We have that z_σ is the sum of all $z_{\langle\sigma,\ \Delta\rangle}$. Hence we can partition the values σ^i into sets σ_Δ of size $z_{\langle\sigma,\ \Delta\rangle}$ respectively.

To define $\mathcal{M}$ on the n-ary function symbol f consider an n-tuple $\langle\sigma_1,\ \ldots,\ \sigma_n\rangle$ of types in S. It suffices to consider a fixed such tuple and define f on all domain tuples of the form $\langle\sigma_1^{i_1}, \ldots, \sigma_n^{i_n}\rangle$. Let D be the set of all such domain tuples (for a fixed tuple $\langle\sigma_1,\ \ldots,\ \sigma_n\rangle$) and let R be the set of all domain elements that are members of any set σ_Δ such that $\Delta(U) = f(\sigma_1, \ldots, \sigma_n)$ for some $U \in \sigma$. Let τ be some type in S such that $f(\sigma_1,\ \ldots,\ \sigma_n) \leadsto \tau$. Both D and R are countable sets so both can be enumerated. Now define f to map each element of D to the corresponding element of R under the given enumeration. The constraints in $D(S)$ ensure that the cardinality of R is no larger than the cardinality of D so every element of R will be the image of some element of D. If for some tuple in D there is no corresponding element of R then f maps that tuple to τ^1.

To prove $\mathcal{M} \models \Sigma$, we first show, by structural induction on a set expression E, that if E occurs anywhere in Σ then $\mathcal{M}[\![E]\!]$ is the set of domain elements σ^i such that σ contains E. If E is a set variable then this follows from the definition of $\mathcal{M}$ on set variables. Suppose E is the negation of an expression E_1 for which the result holds. We show that the result holds for E. Our induction hypothesis tells us that $\mathcal{M}[\![E_1]\!]$ is the set of domain elements whose base Σ-types contain E_1—then of course $\mathcal{M}[\![E]\!]$ is the complement of this set, i.e., the set of domain elements whose base types do not contain E_1. The definition of a Σ-type ensures that every type contains exactly one of E_1 or $\neg E_1$ so the complement of the domain elements whose type contains E_1 is exactly the set of domain elements whose type contains $\neg E_1$.

Now suppose E is the union of two expressions E_1 and E_2 for which the result holds, and show that the result holds for E. Our induction hypothesis tells us that $\mathcal{M}[\![E_1]\!]$ (and respectively, $\mathcal{M}[\![E_2]\!]$) is just the set of all domain members whose base type contains E_1 (respectively, E_2). So $\mathcal{M}((E_1 \cup E_2))$ is the set of all domain members whose base type contains either E_1 or E_2. The definition of a Σ-type ensures that this is exactly the set of all domain members whose base type contains $(E_1 \cup E_2)$, as desired. A similar analysis holds for set intersection expressions.

Now suppose E is the application of the n-ary relation symbol R to expressions $E_1, \ldots, E_n$ for which the result holds. We first show the forward direction—that every domain element whose base type contains $R(E_1, \ldots, E_n)$ is also in $\mathcal{M}[\![R(E_1, \ldots, E_n)]\!]$. Consider a domain element σ^k where σ contains $R(E_1, \ldots, E_n)$. Since the set S is locally consistent there must exist Σ-types $\tau_1, \ldots, \tau_n$ in S containing the $E_1, \ldots, E_n$, respectively, such that $R(\tau_1, \ldots, \tau_n) \rightsquigarrow \sigma$. But then by definition $\mathcal{M}[\![R]\!]$ maps $\langle (\tau_1)^1, \ldots, (\tau_n)^1 \rangle$ to σ^k. But by our induction hypothesis, each element ${\tau_i}^1$ must be in the corresponding $\mathcal{M}[\![E_i]\!]$, so σ^k must be in $\mathcal{M}[\![R(E_1, \ldots, E_n)]\!]$, as desired. For the reverse direction, suppose that a domain element σ^k is in $\mathcal{M}[\![R(E_1, \ldots, E_n)]\!]$.

Because σ^k is in $\mathcal{M}[\![R(E_1, \ldots, E_n)]\!]$, there must be domain elements $(\tau_1)^{(i_1)}$, ..., $(\tau_n)^{(i_n)}$ in $\mathcal{M}[\![E_1]\!], \ldots, \mathcal{M}[\![E_n]\!]$ respectively such that $R(\tau_1, \ldots, \tau_n) \leadsto \sigma$. But, by our induction hypothesis, since each $(\tau_j)^{(i_j)}$ is in $\mathcal{M}[\![E_j]\!]$ we have that each τ_j contains E_j. By the definition of a Σ-type we have that σ contains either $R(E_1, \ldots, E_n)$ or $\neg R(E_1, \ldots, E_n)$. Since $R(\tau_1, \ldots, \tau_n) \leadsto \sigma$, the type σ can not contain $\neg R(\tau_1, \ldots, , \tau_n)$. Hence it must contain $R(E_1, \ldots, E_n)$.

The final case to consider is when E is the application $f(E_1, \ldots, E_n)$ of an n-ary function symbol f to n expressions $E_1, \ldots, E_n$ for which the result holds. Again, we first consider the forward direction—we suppose σ^k is a domain element in $\mathcal{M}[\![f(E_1, \ldots, E_n)]\!]$ and show that σ contains $f(E_1, \ldots, E_n)$. Since σ^k is in $\mathcal{M}[\![f(E_1, \ldots, E_n)]\!]$, it must be the image under f of some tuple of domain elements $\langle(\tau_1)^{(i_1)}, \ldots, (\tau_n)^{(i_n)}\rangle$ which are members, respectively, of $\mathcal{M}(E_1), \ldots, \mathcal{M}(E_n)$. Our induction hypothesis then implies that for each j, E_j is a member of τ_j. When f was defined for tuples of the form $\langle(\tau_1)^{(i_1)}, \ldots, (\tau_n)^{(i_n)}\rangle$, the type σ^k must have been in the range enumeration. There are two ways that this can happen. First, σ^k could be an element of σ_Δ for some Δ containing $f(\tau_1, \ldots, \tau_n)$ in its range. Second, σ could be the "default value" for f on tuples whose base types are $\tau_1, \ldots, \tau_n$. In either case we have $f(\tau_1, \ldots, \tau_n) \leadsto \sigma$. But then, since each E_i is in the corresponding τ_i, $f(E_1, \ldots, E_n)$ must be in σ as desired (by the definition of $\leadsto$ we can not have $\neg f(E_1, \ldots, E_n) \in \sigma$).

It remains to show the reverse direction: we take an arbitrary domain element σ^k such that σ contains $f(E_1, \ldots, E_n)$ and show that this σ^k is a member of $\mathcal{M}[\![f(E_1, \ldots, E_n)]\!]$. For any $\sigma \in S$ we have that z_σ is nonzero. So some $z_{\sigma,\Delta}$ must be nonzero and so some set σ_Δ is nonempty. Since σ contains $f(E_1, \ldots, E_n)$, $\Delta(f(E_1, \ldots, E_n))$ must be some range expression $f(\tau_1, \ldots, \tau_n)$ such that τ_i contains E_i and $f(\tau_1, \ldots, \tau_n) \leadsto \sigma$. But then, by our inductive hypothesis about the E_i, every tuple of domain elements of respective base types $\tau_1, \ldots, \tau_n$ must belong, respectively, to $\mathcal{M}[\![E_1]\!], \ldots, \mathcal{M}[\![E_n]\!]$. This is the entire domain enumeration for f on $\tau_1, \ldots, \tau_n$, so the entire range enumeration for f on $\tau_1, \ldots, \tau_n$ must be contained in $\mathcal{M}[\![f(E_1, \ldots, E_n)]\!]$. But this range enumeration must include σ^k since it is a member of σ_Δ and Δ contains $f(\tau_1, \ldots, \tau_n)$ in its range.■

The above lemma shows that satisfiability for nonrecursive Tarskian constraints can be reduced in nondeterministic exponential time to satisfiability of a system of diophantine constraints where the variables range over positive integers plus ∞. We can eliminate ∞ by nondeterministically guessing which variables are infinite and folding this guess into the constraints. More specifically, we substitute ∞ in for the variables we guess to be infinite and check that in any equality with an infinity in the left hand side there is also an infinity on the right hand side. We then remove all inequalities involving infinity. This produces a set of diophantine constraints over positive integers. These constraints have the following form.

> **Definition:** A set of Diophantine inequalities $\{ p_i(x_1, \ldots, x_n) \leq q_i(x_1, \ldots, x_n) \mid 1 \leq i \leq m \}$ between polynomials p_i and q_i over non-negative integer variables $x_1, \ldots, x_n$ is *semi-linear* if every p_i is linear, and every q_i is either linear or is a product of variables.

The size of the system of diophantine constraints generated by a set Σ of nonrecursive Tarskian constraints is exponential in $|\Sigma|$. The following theorem shows that satisfiability of semi-linear diophantine constraints can be determined in nondeterministic exponential time. We conjecture that satisfiability of semi-linear diophantine constraints is actually in NP.

> **Semi-Linear Decidability Theorem:** The problem of determining the satisfiability of a semi-linear set of Diophantine inequalities is solvable in non-deterministic exponential time.

Proof: Consider a semi-linear set of m Diophantine inequalities over n variables where the largest constant which appears has b bits. Each inequality is either linear or has a right hand side that is the product of two variables. The linear inequalities can be converted into an equisatisfiable set of Diophantine equations $Ax = B$ by introducing new "slack" variables. Call a variable x_i *bounded* in $Ax = B$ if there exists a finite upper bound on the value of x_i over all *rational* solutions to $Ax = B$. We can use linear programming (over the rationals) to determine which variables are bounded. An analysis of the maximum possible upper bound that can be imposed by a system of linear constraints shows that the binary representation of the value of a bounded variable can contain at most $O(bn \log n)$ bits. Our nondeterministic procedure can now guess the values of the bounded variables. We can then replace each bounded variable by the guessed value giving a simplified problem. In substituting in the guesses, some of the nonlinear constraints become linear and must be added to the resulting linear sub-problem, yielding new linear and nonlinear subproblems in fewer variables. We repeat this process until either all variables have been guessed or all the variables in the resulting linear problem are unbounded. If at some point the guesses lead to a linear problem which is unsolvable over the rationals then all variables are bounded and values for them are guessed. Of course in practice the procedure would simply terminate when such an inconsistency is found. Once all variables in the remaining linear problem are unbounded we determine the solvability of $Ax = B$ over the nonnegative integers. If the residual problem $Ax = B$ is solvable over the nonnegative integers then accept the original semi-linear problem as solvable. Otherwise we fail.

We must show that this procedure is correct and that it terminates in nondeterministic exponential time. First we consider the running time analysis. The number of bits in the bounded variables can grow at most exponentially in the number of iterations of the procedure. One can show that after k guesses the largest constant in the residual linear problem has at most $O(b(cn \log n)^{k+1})$ bits for some constant c. Since the number of guesses is bounded by n, we get an exponential upper bound on the size of the numbers appearing in the sequence of linear problems examined by the procedure. Since all the linear programming operations over the rationals can be done in polynomial time, and since integer programming is in NP, we get a nondeterministic exponential running time.

We must now show that if the procedure accepts a semi-linear system of constraints then that constraint set is solvable. If the procedure accpets then there exists a residual linear problem $Ax = B$ plus a residual set of nonlinear constraints. It suffices to show that this residual semi-liear problem is solvable over the nonnegative integers. Since the procedure accepts, there exists a nonnegative integer solution β to $Ax = B$. It is a fact of linear programming that if all variables are unbounded then there must be a nonnegative rational solution α to $Ax = 0$ such that all components of α are nonzero. We can assume without loss of generality that α is integral because any nonintegral α

can be made integral by multiplying by an appropriate constant. The vector $\beta + c\alpha$ is a solution to $Ax = B$ for any c. For sufficiently large c this vector also satisfies all nonlinear constraints because the nonlinear expressions must eventually grow faster than the linear expressions.■

Finally, we can combine the above results to get the following.

> **Theorem:** Satisfiability of nonrecursive Tarskian set constraints is decidable in nondeterministic doubly exponential time.

9 Recursive Constraints without Functions

In this section we consider recursive Tarskian set constraints not involving function symbols. We show that the satisfiability problem for these set constraints is linear time equivalent (under Turing reductions) to satisfiability for the propositional μ-calculus.

To better understand the relationship between Tarskian set constraints without functions, PDL, and the μ-calculus we define some additional terminology.

> **Definition:** A relational model is a model that does not provide any interpretation for function symbols. Let x be an element of the domain of a relational model $\mathcal{M}$. The *inverse image tree* of x in $\mathcal{M}$, denoted $T(\mathcal{M}, x)$, is a possibly infinite tree where each node of the tree is labeled either with an element of $\mathcal{M}$ or with a tuple $\langle R, y_1, \ldots, y_n\rangle$ where R is a relation symbol and each y_i is a member of the domain of $\mathcal{M}$. The root node is labeled with x. The children of any node are generated from the label of that node in the following way. If the node is labeled with an element z then for each tuple $\langle R, y_1, \ldots, y_n\rangle$ such that $\langle y_1, \ldots, y_n, z\rangle \in \mathcal{M}(R)$ there is a child of z with label $\langle R, y_1, \ldots, y_n\rangle$. Each node labeled with a tuple $\langle R, y_1, \ldots, y_n\rangle$ has n children with labels $y_1, \ldots, y_n$ respectively.

The *inverse-image substructure* of $\mathcal{M}$ rooted at x, denoted $I(\mathcal{M}, x)$ is a model whose domain consists of the elements that occur as labels in the inverse image tree $T(\mathcal{M}, x)$ and which interprets each relation symbol R as the restriction of $\mathcal{M}(R)$ to the domain of $I(\mathcal{M}, x)$.

For example, if $\mathcal{M}$ consists of the natural numbers and the single relation $>$, where $>(C)$ is the set of numbers greater than some member of C, then the inverse image substructure $I(\mathcal{M}, n)$ is the set of numbers less than or equal to n with the relation $>$ being the restriction of $>$ to these points. In this case $I(\mathcal{M}, n)$ is finite, although that need not be true in general.

Lemma: For any point element x in the domain of a model $\mathcal{M}$ and any set expression C involving only relation symbols interpreted by $\mathcal{M}$, we have that $x \in \mathcal{M}[\![C]\!]$ if and only if $x \in I(\mathcal{M}, x)[\![C]\!]$.

Proof: We can determine if $x \in \mathcal{M}[\![C]\!]$ by treating C as a predicate to be run on the inverse image tree of x. To evaluate $\neg C$ on x we evaluate C on x and then negate the resulting truth value. Other Boolean operations are handled similarly. The predicate $R(C_1, \ldots, , C_n)$ is true of an element z if the inverse image tree rooted at z contains a child labeled with the tuple $\langle R, y_1, \ldots, , y_n \rangle$ such that each predicate C_i is recursively true of the corresponding element y_i. A μ-set $\mu X C[X]$ is true of an element x if there is some ordinal α such that $C^\alpha[F]$ is true of x where $C^0[F]$ is F and $C^\beta[F]$ is true of x if there is some $\alpha < \beta$ such that $C[C^\alpha[F]]$ is true of x. We can show by transfinite induction on α that the truth of $C^\alpha[\mathbf{F}]$ on x is determined by inverse image tree of x. So for any C we have that the truth of C on x depends only on the inverse image tree of x. The inverse image tree of x in $I(\mathcal{M}, x)$ is the same as the inverse image tree of x in $\mathcal{M}$.

Now we show that the satisfiability of any system of Tarskian set constraints not involving function symbols can be reduced to a set of questions about the satisfiability of individual recursive set expressions. Note that the following definition also handles function symbols.

Definition: Let Σ be a set of positive Tarskian set constraints. We define $C[\Sigma]$ to be

$$\mu X.\ C \cup R_1(X,\ , \ldots,\ X) \cup \ldots \cup R_k(X,\ \ldots,\ X)$$

where X is a set variable not occurring in Σ, the relation symbols $R_1, \ldots R_k$ are all relation symbols of arity greater than 1 and all function symbols of arity greater than 0 (in either case the number of arguments taken in class expressions is at least one) and C is the "counter example" formula

$$(\neg U_1 \cap W_1) \cup \ldots \cup (\neg U_n \cap W_n)$$

where Σ is the set of constraints $W_1 \subseteq U_1, \ldots, W_n \subseteq U_n$.

One should think of the set expression $C[\Sigma]$ as a predicate to be be run on the inverse image tree of a point x. This predicate is true of x if and only if some node in the inverse image tree of x is labeled with a counter example to the (positive) constraints in Σ. This observation yields the following lemma:

Lemma: Let Σ be a set of positive constraints not involving function symbols and let $\mathcal{M}$ be a relational model. If $x \in \mathcal{M}[\![\neg C[\Sigma]]\!]$ then $I(\mathcal{M}, x) \models \Sigma$.

Note that the above lemma requires that Σ not contain function symbols. If Σ contained function symbols then a natural definition of $I(\mathcal{M}, x)$ would be forced to interpret function symbols as partial functions because the domain of $I(\mathcal{M}, x)$ may contain input tuples such that the corresponding output value is not in the inverse image domain. If we add an element to $I(\mathcal{M}, x)$ to serve as a "default value" for the function symbols then we may introduce a point violating the subset constraints in Σ.

The above lemma yields the following theorem.

Definition: We say that a Tarskian set expression C is *satisfiable* if there exists some model $\mathcal{M}$ and some element x of the domain of that model such that $x \in \mathcal{M}[\![C]\!]$.

Lemma: Let Σ be a set of Tarskian set constraints not involving function symbols. Let Σ' be the positive set constraints in Σ and let $U_1 \not\subseteq W_1, \ldots, U_k \not\subseteq W_k$ be the negative constraints in Σ. Let $C_1, \ldots, C_k$ be the set expressions such that C_i is $U_i \cap \neg W_i \cap \neg C[\Sigma']$. Σ is satisfiable if and only if each C_i is a satisfiable set expression.

Proof: Suppose that each C_i is satisfied by a point x_i in a model $\mathcal{M}_i$. Then by the preceding lemma each inverse image structure of the form $I(\mathcal{M}_i, x_i)$ is a model of the positive constraint set Σ'. The disjoint union of all inverse image structures $I(\mathcal{M}, x_i)$ satisfies both the positive constraint set Σ' and all the negative constraints in Σ.

We have now reduced the problem of satisfiability of Tarskian constraints without function symbols to the satisfiability problem for individual Tarskian set expressions. This reduction requires that the set constraints not involve function symbols. However, the satisfiability problem for individual Tarskian set expressions is insensitive to the presence of function symbols. We will now show that satisfiability problem for recursive Tarskian set expressions *with arbitrary function symbols* is essentially equivalent to the satisfiability problem for the propositional μ-calculus.

There is actually a fairly wide variety of formulations of the μ-calculus which all turn out to be equivalent in the sense that the corresponding satisfiability problems for individual set expressions are linear time equivalent. All of these calculi include set variables, Boolean operations on sets, and least fixed point expressions of the form $\mu X.C[X]$ where $C[X]$ is a set expression involving X. As we will show below, the following set calculi have essentially equivalent satisfiability problems for individual set expressions.

The Tarskian μ-calculus: This consists of the Tarskian set expressions defined in section 2 (with recursion and arbitrary function symbols).

The Herbrand μ-calculus: The set expression of this language denote sets of possibly infinite terms constructed from a given set of constructor function symbols. This calculus allows set expressions of the form $f(C_1, \ldots, C_n)$ where f is a constructor function symbol of n arguments and each C_i is a set expression. The expression $f(C_1, \ldots, C_n)$ denotes the set of (possibly infinite) terms of the form $f(t_1, \ldots, t_n)$ where each t_i is a term in the set denoted by C_i.

The Herbrand μ-calculus with one constant symbol and one binary function symbol.

The Tarskian μ-calculus with one monadic function symbol and no relation symbols.

The Dynamic μ-calculus: This is the traditional "propositional μ-calculus". The models can be taken to be first order structures interpreting a set of binary and monadic predicate symbols. In addition to the standard way of constructing set expressions the dynamic μ-calculus allows set expressions of the form P where P is a monadic predicate and $<R>\ C$ where R is a binary relation. The latter expression denotes the set $\{x : \ \exists y \in C : \ \ R(x, y)\}$.

The Herbrand calculus seems most natural for understanding satisfiability algorithms for set expressions in these calculi. The existing decision procedures are based on an implicit translation of the dynamic μ-calculus into the Herbrand calculus [Streett and Emerson, 1989], [Emerson and Jutla, 1988], [Safra, 1988]. The Herbrand calculus is based on the Herbrand universe of possibly infinite terms over a given set of function symbols. This would seem to indicate a relationship between the Herbrand calculus and Herbrand set constraints. However, in traditional Herbrand set constraint problems we are concerned with the existence of certain *sets* of Herbrand terms while here we are concerned with the existence of a single (possibly infinite) term satisfying given constraints.

There are many interesting examples of term sets definable in the Herbrand μ-calculus. The expression $\mu X.a \cup f(X)$ is the set of all finite terms which are some number of applications of f to a. We let $\nu X.C[X]$, a greatest fixed point expression, be an abbreviation for $\neg\mu X.\neg C[\neg X]$. The expression

$\nu X.f(X)$ denotes a singleton set containing the infinite term $f(f(f(\ldots)))$. We will abbreviate this expression as f^ω. Another interesting example is $\mu X.g^\omega \cup f(X) \cup g(X)$. This is the set of infinite terms constructed from monadic function symbols f and g which have only finitely many occurrences of f. One can similarly define the set of infinite terms constructed from f and g that have only finitely many occurrences of g. Any satisfiability testing procedure must be capable of detecting the fact that the intersection of these two term sets is empty. It is known that the Herbrand μ-calculus defines exactly those term sets definable by Rabin tree automaton, or alternatively by formulas of SnS (the second order theory of n successors) [Emerson and Jutla, 1991].

We will give satisfiability preserving reductions between set expressions in these calculi establishing linear time equivalence of the set expression satisfiability problems. We give the reductions only — the proof that these reductions are satisfiability preserving is left to the reader. There is one subtlety in the reduction from the Tarskian μ-calculus to the Herbrand μ-calculus. It is easy to show that this reduction is satisfiability preserving provided all satisfiable Tarskian set expressions can be satisfied by a countable model. To see that this is true consider a Tarskian set expression C and a pair $\langle x, \mathcal{M}\rangle$ satisfying C, i.e., such that $x \in \mathcal{M}[\![C]\!]$. Think of C as a predicate and consider running the predicate C on the inverse image tree $T(x, \mathcal{M})$. Since C is true of x (in $\mathcal{M}$) there must exist a successful execution of C on the tree $T(x, \mathcal{M})$. One can show that a successful execution need only involve a finite number of children of any given node in $T(x, \mathcal{M})$. Hence the number of nodes involved in the execution is countable. By restricting $T(x, \mathcal{M})$ to the set of nodes actually involved in the successful execution of C we get a countable model of C.

In all the following translations we have that $T(X)$ is X, $T(\neg C)$ is $\neg T(C)$, $T((C_1 \cup C_2))$ is $(T(C_1) \cup T(C_2))$, and $T(\mu X.C[X])$ is $\mu X.T(C[X])$. In all calculi we use F as an abbreviation for $\mu X.X$ and T as an abbreviation for $\neg F$.

The Tarskian calculus to the Herbrand calculus.

$$T(R(C_1, \ldots, C_n)) = \mu X f_R(T(C_1), \ldots, T(C_n)) \cup g(\mathbf{T}, X) \cup g(X, \mathbf{T})$$

where X is a fresh set variable, f_R is a function symbol associated with the function or relation R, g is a new function symbol which intuitively allows branching in the inverse image tree, and $\mathbf{T}$ is the set $Z \cup \neg Z$ for an arbitrary set variable Z.

Herbrand calculus to Herbrand calculus with one constant a and one binary function f

$$T(g(C_1,\ \ldots,\ C_{n-1},\ C_n)) = f(t_g, f(T(C_1),\ f(\ldots,\ f(T(C_{n-1}), T(C_n)))))$$

where t_g is a ground term built from a and f such that t_g is different from t_h for every pair of distinct function symbols g and h.

Herbrand calculus with one constant symbol a and one binary function f to the Tarskian calculus with one monadic function symbol s

$$T(a) = Q$$

$$T(f(C_1,\ C_2)) = \neg s(P \cap R) \cap s(P \cap (T(C_1))) \cap s(R \cap T(C_2))$$

where $Q = \neg s(X \cup \neg X)$, $P = s(Q)$ and $R = s(P)$.

Tarskian calculus with one monadic function symbol s to the dynamic calculus

$$T(s(C)) = <R>\ T(C)$$

Dynamic calculus to the Tarskian calculus

$$T(<R>\ C) = R(T(C))$$

It is known that satisfiability of set expressions in the dynamic μ-calculus is complete for exponential time [Streett and Emerson, 1989], [Emerson and Jutla, 1988], [Safra, 1988]. Hence the reduction from satisfiability of Tarskian set constraints without function symbols to satisfiability in the dynamic μ-calculus establishes the following theorem.

> **Theorem:** Satisfiability for Tarskian set constraints without function symbols is exponential time complete.

10 Recursive Constraints with Functions

In this section we show that satisfiability for full Tarskian set constraints (with recursion and arbitrary functions) is undecidable. The proof is by a reduction of Hilbert's tenth problem. The proof uses only set variables, constants, monadic functions, set unions and intersections (no complementation), and a single level of μ quantification.

> **Theorem:** Satisfiability for Tarskian set constraints with constants and monadic functions is undecidable.

Proof: Let Σ be a set of constraints of the form $n = 1$, $n = p + q$ or $n = pq$ where n, p and q range over nonnegative integers. It follows from the undecidability of Hilbert's tenth problem that satisfiability for such systems of constraints is undecidable. We reduce the diophantine constraint set Σ to a set $T(\Sigma)$ of Tarskian set constraints as follows.

For each natural number variable n occurring in Σ we introduce a set variable X_n with the intention that the cardinality of X_n represents the value of n. For set expressions C and W we will use $C = W$ as an abbreviation for the two constraints $C \subseteq W$ and $W \subseteq C$. We will also use $|C| \leq |W|$ as an abbreviation for $C \subseteq f(W)$ where f is a fresh monadic function symbol. We will use $|C| = |W|$ as an abbreviation for $|C| \leq |W|$ and $W| \leq |C|$. For any monadic function symbol s and class expression C we let $s^*(C)$ be an abbreviation for $\mu W.\ C \cup s(W)$, i.e., the set of things that can be gotten by applying s zero or more times to an element of C. For each variable n in Σ we introuce a constant symbol c_n and monadic function symbol $_n$ and add the constraints

$$X_n = s_n^*(c_n)$$

$$c_n \subseteq s_n(s_n^*(c_n)).$$

The first constraint states that X_n is the set containing c_n and all its transitive successors under s_n. The second constraint states that c_n is the successor of some element of $s_n^*(c_n)$ and therefore that the set X_n forms a loop under the successor function s_n. This implies that X_n is a finite set but does not otherwise constrain its cardinality. We now need to impose the constraints given in Σ.

If Σ contains the constraint $n = 1$ then $T(\Sigma)$ contains the constraint $X_n = c_n$. If Σ contains $n = p + q$ then we add the constraints

$$X_n = U \cup W$$

$$|X_p| = |U|$$
$$|X_q| = |W|$$
$$U \cap W \subseteq F$$

to $T(\Sigma)$ where C, U, and W are fresh set variables and F is the set expression $\mu X.\ X$. It remains only to express product constraints.

To handle the product case we use the notation $\forall x \in f^*(c) \;\; x = C[x]$ as an abbreviation for

$$f^*(c) = \mu X.(c \cap C[c]) \cup (f(X) \cap C[f(X)])$$

For example, $\forall x \in f^*(c) \quad x = g(f(x))$ states that g is the inverse of f on the set $f^*(c)$. More generally, if there is only one occurance of X in $C[X]$, and $C[X]$ is constructed purely from X and function symbols, then $\forall x \in f^*(c) x = C[x]$ has the obvious intended meaning. Suppose Σ contains $n = pq$. We add the following constraints to $T(\Sigma)$.

1. $X_n = f^*(c)$
2. $c \subseteq f(f^*(c))$
3. $X_p = g^*(c)$
4. $c \subseteq g(g^*(c))$
5. $X_q = h^*(c)$
6. $c \subseteq h(h^*(c))$
7. $f^*(c) = \mu X \; c \cup g(X) \cup h(X)$

8. $\forall x \in f^*(c) \;\; x = g'(g(x))$

9. $\forall x \in f^*(c) \;\; x = h'(h(x))$

10. $\forall x \in f^*(c) \;\; x = g'(h'(g(h(x))))$

11. $g^*(c) \cap h^*(c) = c$

Where c is a fresh constant and f, g, h, g' and h' are fresh monadic function symbols. Constraints 2, 4, and 6 imply that $f^*(c)$, $g^*(c)$, and $h^*(c)$ are all "loops". Constraint 7 implies that g and h are both functions mapping $f^*(c)$ into $f^*(c)$. Constraints 8, and 9 imply that g', and h' are inverses of g and h respectively on the set $f^*(c)$. Since both g and h are invertible they must both be bijections from $f^*(c)$ to itself. This implies that the inverses g' and h' are also bijections. Condition 10 implies that g and h commute on $f^*(c)$, i.e., $f(g(x))$ equals $g(f(x))$. Now consider $g^n(h^*(c))$. Since g is bijective, g^n is bijective. Note that $h(g^n(x))$ equals $g^n(h(x))$. So the mapping g^n is a bijection which "preserves h structure". Hence the set $g^n(h^*(c))$ is an h-loop with the same cardinality as $h^*(c)$. Since sets of the form $g^j(h^*(c))$ are h-loops they are either equal or disjoint. Suppose $g^j(h^*(c)) = g^k(h^*(c))$. Applying $(g')^j$ to both sides we $h^*(c) = g^{k-j}(h^*(c))$. This implies that $g^{k-j}(c)$ must be in $h^*(c)$ and hence by condition 11 above we have $g^{k-j}(c) = c$. But this implies that k equals j mod $|g^*(c)|$. Hence for $k \neq j$ mod $|g^*(c)|$ we have $g^j(h^*(c))$ is disjoint from $g^k(h^*(c))$. Since all these sets are of size $|h^*(c)|$ we have $|f^*(c)| = |g^*(c)||h^*(c)|$. ■

11 Conclusions

The semantics of modal and temporal logics are usually described in terms of Kripke structures. However, a Kripke structure is easily seen to be equivalent to a classical first order structure whose domain is the set of possible worlds. Given this view of Kripke structures, concept languages, temporal and modal logics, and Tarskian set expressions can all be viewed as restricted languages for describing subsets of a first order domain. Each of these logics is defined by basic primitives for forming sets. Tarskian set expressions are perhaps the oldest and most natural but least widely studied of these formalisms. Of the various fragments of Tarskian set expressions studied here, only recursive expressions without functions seems to correspond to an existing widely studied logic — the propositional μ-calculus. Our main results are that satisfiability for nonrecursive Tarskian set constraints is decidable in nondeterministic doubly exponential time and that satisfiability in full Tarskian set constraints (with recursion and arbitrary function symbols) is undecidable.

This paper leaves several open problems. First, we conjecture that satisfiability for nonrecursive Tarskian set constraints (with arbitrary function symbols) is solvable in nondeterministic exponential time. Our attempts to prove this have been based on the reduction of the problem to semi-linear diophantine equations. We conjecture that the satisfiability problem for semi-linear diophantine equations is in NP although our current best decision procedure only gives a nondeterminisitic exponential time upper bound.

The second major open problem concerns recursive Tarskian set constraints with limited function symbols. The main difficulty in constructing decision procedures for satisfiability of Tarskian set constraints with arbitrary function symbols is the ability to express diophantine constraints. The ability to express diophantine constraints requires the presence of both constant symbols and function symbols of arity greater than zero. Without both functions and constants there is no clear approach to showing undecidability. Nor is there a clear approach to finding a decision procedure since the reduction to the propositional μ-calculus appears to fail. We can not expect a simple decision procedure because any such procedure could be used to solve satisfiability in the classical μ-calculus.

References

[Aiken and Wimmers, 1994] A. Aiken and E. Wimmers. Soft typing with conditional types. In *ACM Symposium on Principles of Programming Languges*, pages 163–173, January 1994.

[Bachmair *et al.*, 1993] Leo Bachmair, Herald Ganzinger, and Uwe Waldmann. Set constraints are the monadic class. In *Proceedings, Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 75–83, Montreal, Canada, 19–23 June 1993. IEEE Computer Society Press.

[Brachman and Schmolze, 1985] Ronald Brachman and James Schmolze. An overview of the kl-one knowledge representation system. *Computational Intelligence*, 9(2):171–216, 1985.

[Calvanese *et al.*, 1994] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class-based representation formalisms. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, pages 109–119, May 1994.

[Donini *et al.*, 1991] Francesco Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. In *Proceedings of KR91*, pages 151–162. Morgan Kaufmann Publishers, 1991.

[Downing and Gallier, 1984] William Downing and Jean H. Gallier. Linear time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 1(3):267–284, 1984.

[Emerson and Jutla, 1988] E.A. Emerson and C. S. Jutla. The complexity of tree automata and modal logics of programs. In *Proc. 29th Ann. IEEE Symp. on Foundations of Computer Science* [1988], pages 328–337.

[Emerson and Jutla, 1991] E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus, and determinancy. In *Proc. 32nd Ann. IEEE Symp. on Foundations of Computer Science*, pages 368–377, 1991.

[Fischer and Ladner, 1978] M.J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Comput. System Sci.*, 18(2):194–211, 1978.

[Giacomo and Lenzerini, 1994a] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and dynamic logic. In *12th Nat. Conf. on Artificial Intelligence*, pages 205–212, August 1994.

[Giacomo and Lenzerini, 1994b] G. De Giacomo and M. Lenzerini. Concept languages with number restrictions and fixed points and its relationship with mu-calculus. In *11th European Conf. on Artificial Intelligence*, pages 356–360, August 1994.

[Givan and McAllester, 1992] Robert Givan and David McAllester. New results on local inference relations. In *Principles of Knolwedge Representation and Reasoning: Proceedings of the Third International Conference*, pages 403–412. Morgan Kaufman Press, October 1992. internet file ftp.ai.mit.edu:/pub/dam/kr92.ps.

[Goldblatt, 1989] Robert Goldblatt. Varieties of complex algebras. *Annals of Pure and Applied Logic*, 44:173–242, 1989.

[Heintze and Jaffar, 1990] H. Heintze and J. Jaffar. A finite presentation theorem for approximating logic programs. In *ACM Symposium on Principles of Programming Languges*, pages 197–209, January 1990.

[IEE, 1988] IEEE. *Proc. 29th Ann. IEEE Symp. on Foundations of Computer Science*, 1988.

[Jònsson and Tarski, 1951] B. Jònsson and A. Tarski. Boolean algebras with operators. part i. *Amer. J. Math*, 73:891–939, 1951.

[Jònsson and Tarski, 1952] B. Jònsson and A. Tarski. Boolean algebras with operators. part ii. *Amer. J. Math*, 74:127–167, 1952.

[Kozen, 1983] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[Kozen, 1993] D. Kozen. Logical aspects of set constraints. In *Proc. 1993 Conf. Computer Science Logic (CSL93)*. Eur. Assoc. Comput. Sci. Logic, September 1993.

[Lewis, 1980] Harry R. Lewis. Complexity results for classes of quantificational formulas. *J. of Comp. Syst. and Syst. Sci.*, 21:317–353, 1980.

[McAllester and Givan, 1992] D. McAllester and R. Givan. Natural language syntax and first order inference. *Artificial Intelligence*, 56:1–20, 1992. internet file ftp.ai.mit.edu:/pub/dam/aij1.ps.

[McAllester and Givan, 1993] D. McAllester and R. Givan. Taxonomic syntax for first order inference. *JACM*, 40(2):246–283, April 1993. internet file ftp.ai.mit.edu:/pub/dam/jacm1.ps.

[Pratt, 1976] V. Pratt. Semantical considerations on floyd-hoare logic. In *FOCS76*, pages 109–121, 1976.

[Pratt, 1980] V. R. Pratt. A near-optimal method for reasoning about actions. *J. Comput. System Sci.*, 20(2):231–254, 1980.

[Safra, 1988] S. Safra. On the complexity of ω-automaton. In *Proc. 29th Ann. IEEE Symp. on Foundations of Computer Science* [1988], pages 319–327.

[Schmidt-Schaub and Smalka, 1991] M. Schmidt-Schaub and G. Smalka. Attributive concept descriptions with complements. *Artificial Intelligence*, 47:1–26, 1991.

[Streett and Emerson, 1989] R. S. Streett and E. A. Emerson. An automaton theoretic decision procedure for the propositional μ-calculus. *Information and Computation*, 81:249–264, 1989.