UNIVERSITY OF CATANIA
Department of Mathematics and Computer Science

SET THEORY FOR KNOWLEDGE REPRESENTATION

Cristiano Longo

A dissertation submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Tesi presentata per il conseguimento del titolo di "Dottore di Ricerca in Informatica" (XXIV ciclo)

Coordinatore e Tutor
Chiarissimo Prof. Domenico Cantone

Contents

Chapter 1

Introduction

During the last century Set Theory played a central role in the development of *modern* mathematics, as it provided a single foundation for the diverse areas of this discipline. For example, the intuitive content of geometry, arithmetic, and analysis has been re-expressed in set-theoretic terms in entirely precise formal fashion.

The decision problem in set theory has been intensively investigated in the last decades, thus giving rise to the novel research field of *Computable Set Theory*, devoted to study the decision problem for fragments of set theory (see [11, 15] for a thorough account of the state-of-the-art until 2001).

The motivation which initially animated this research stream was the design and implementation of an interactive proof checker, envisaged by Jackob T. Schwartz[1] in [41], supposed to accept sequences of logical formulae, such that any formula in the sequence follows logically from earlier formulae. In addition, such a system should include also an *inferential core*, based on the very expressive formalism of set theory, to handle elementary inferential steps, in order to keep at a reasonable level the mass of details in proofs.

Over the years, decision procedures or proofs of undecidability have been provided for several quantified and unquantified fragments of set theory. It is to be noticed, however, that several decision procedures found so far are not practical at all and their interest is limited to the foundational purpose of identifying the boundary between the decidable and the undecidable in set theory, while the most efficient decision procedures devised in this context have been implemented in the inferential core of the proof verifier ÆtnaNova/Referee, described in [16, 33, 42].

The first unquantified sublanguage of set theory that has been proved decidable is *Multi-Level Syllogistic* (in short MLS, cf. [19]). MLS involves the set predicates $\in$, $\subseteq$, $=$, the Boolean set operators $\cup$, $\cap$, $\setminus$, and the connectives of propositional logic (see Section 2.2 for the precise definitions of MLS syntax and semantics). Subsequently, several extensions of MLS with various combinations of operators (such as singleton, powerset, unionset,

[1] January 9, 1930 – March 2, 2009.

etc.) and predicates (on finiteness, transitivity, etc.) have been proved to have a solvable satisfiability problem.

Also, some extensions of MLS with various map[2] constructs have been shown to be decidable. In [19] a two-sorted extension of MLS is presented, where the two sorts indicate respectively set variables and map variables, and which contains the domain, range, direct and inverse image operators on maps. Additionally, this language allows assertions meaning that a map term is single-valued (i.e. it is a function) or bijective. However, the decision procedure proposed there had a double exponential time complexity. We mention also a one-sorted version of the language considered in [19], further extended with map evaluation (cf. [17]). In this case, since there is no distinction between set and map variables, maps can be combined with the Boolean set operators as well. However, the language in [17] does not allow a predicate $\mathsf{is\_map}(x)$, asserting that x is a collection of pairs. Therefore, for instance, a predicate of type $\mathsf{Inverse}(f, g)$ expresses only that g is an inverse of f, up to non-pair elements, so that $\mathsf{Inverse}(f, g)$ and $\mathsf{Inverse}(f, g')$ do not imply that $g = g'$, but only that g and g' contain the same pairs (namely the inverse of the pairs contained in f). Despite the peculiarity of such semantics, the decision procedure given in [17] has a nondeterministic exponential time complexity.

Concerning *quantified* fragments, of particular interest to us is the restricted quantified fragment of set theory $\forall_0$, viewed in Section 2.3, which has been proved to have a decidable satisfiability problem in [6]. We recall that $\forall_0$-formulae are propositional combinations of restricted quantified prenex formulae $(\forall y_1 \in z_1) \cdots (\forall y_n \in z_n)p$, where p is a Boolean combination of atoms of the forms $x \in y$, $x = y$, and no z_j is a y_i (i.e., nesting among quantified variables is not allowed). The same paper considered also the extension with another sort of variables representing *single-valued* maps, the map domain operator, and terms of the form $f(t)$ (representing the value of the map f on a function-free term t). However, neither one-to-many, nor many-to-one, nor many-to-many relationships can be represented in this language. We observe that the $\forall_0$-fragment is very close to the undecidability boundary, as shown in [37]. In fact, if nesting among quantified variables in prenex formulae of type $(\forall y_1 \in z_1) \cdots (\forall y_n \in z_n)p$ are allowed and a predicate stating that a set is an unordered pair is also admitted, then it turns out that the satisfiability for the resulting collection of formulae is undecidable.

In this thesis we study the decision problem for three novel quantified fragments of set theory, which we denote as the $\forall^\pi$-family of languages. The expressive power of languages of this family is measured in terms of set-theoretical constructs they allow to express. In addition, these languages can be profitably employed in *knowledge representation*, since they allow to express a large amount of *description logic* constructs (cf. Section 3).

[2]According to [43], we use the term 'maps' to denote sets of ordered pairs.

1.1 Knowledge Representation

As mentioned before, set theory allows to define in a elegant and precise way all common mathematical notions. Furthermore (and consequently) it can be used to represent knowledge about several domains in a simple and formal fashion. For this reason, we investigated the application of Computable Set Theory to *knowledge representation*, the field of Artificial Intelligence which deals with formalisms that are both epistemiologically and computationally adequate to represent knowledge about different domains.

Description Logics (see [2] for an introduction) are a family of logic based formalisms widely used in knowledge representation. Several results and decision procedures devised in this context have been profitably employed in the area of the Semantic Web (cf. [5]). For example, the description logic $\mathcal{SROIQ}$, described in [24], underpins the semantic web language OWL 2. Description Logics are logic-based formalisms that allow to represent knowledge about a domain of interest in terms of *concepts*, which denote sets of elements, *roles*, which represent relations between elements, and *individuals*, that denote domain elements. Each description logic is mainly characterized by a set of *constructors*, which allow to form complex terms starting from concept, role and individual names. Finally, a description logic *knowledge base* is a finite set of constraints on the domain structure. The most common description logic constructs are listed in Tables 3.1, 3.2, and 3.3. The description logic framework will be discussed in details in Chapter 3.

1.1.1 SWRL Rules

Other approaches to knowledge representation are those related with *Horn-Style rules*. SWRL rules are a simple form of Horn-style rules which were proposed in [26] with the aim of increasing the expressive power of description logics. It must be noticed that extending description logics with SWRL rules in general leads to undecidability. In [32] this issue has been overcome by restricting the applicability of rules to a finite set of named individuals. Another approach, studied in [29], consists in restricting the set of allowed rules to those which can be *internalized*, i.e. which can be converted into description logic statements.

1.2 Metamodeling

Description logics strictly separate the *conceptual layer*, which defines classes and properties, from the *data layer*, which contains domain objects. As a consequence, such framework does not allow *metamodeling*, namely the ability to define *meta-concepts* (i.e., concepts containing other concepts and roles) or *meta-roles* (i.e., relationships among concepts or among roles). We clarify the notion of metamodeling by way of an example originally presented

in [46]. Let us consider the "IUCN Red List[3] of endangered species," and let *Eagle* be a species in this list. Furthermore, let *Harry* be an eagle. The membership relations among the three items *RedList*, *Harry*, and *Eagle* cannot be accurately modeled in description logics, since *Eagle* should be simultaneously a concept, as it contains *Harry* as member, and an individual, since it is a member of *RedList*. Furthermore, there is no (intuitive) way to arrange a description logic knowledge base in such a way that an automated agent can infer that *Harry* cannot be hunted from the fact that it is a member of *RedList*.

The lack of metamodeling features is sometimes perceived as a crucial limitation of the description logics framework. For example, metamodeling was one of the original goals of the Semantic Web Language, and the earlier version of this language, called OWL Full, was expressive enough to include features of this sort. However, it turned out that this language is undecidable, as proved in [31], where a decidable restriction of OWL Full relative to two alternative semantics, the *contextual* and the *HiLog* semantics, has been proposed. Specifically, in the contextual semantics (also known as *punning*), identifiers are interpreted as individuals or as concepts, depending on the *context*, but the individual and the corresponding concept are treated as entirely independent, so that some of the expected conclusions can not be drawn (for example from $Eagle(Harry), \neg Aquila(Harry)$ it does not descend that the individuals associated to *Eagle* and *Aquila* must be distinct). The HiLog semantics is not affected by this issue. However, it has some consequences which may be considered counterintuitive, as they contrast with the Zermelo-Fraenkel axiomatization of set theory (which instead underpins the family of $\forall^\pi$ languages). For example, a knowledge base $\{A(B), B(A)\}$ is consistent under this semantics, in contrast with the regularity axiom; also, we may have two distinct domain items whose concept and role extensions coincide, thus contradicting the extensionality axiom.

Another strategy to enforce Semantic Web with metamodeling capabilities consists in adding on top of a *domain* knowledge base a separate one, which contains meta-knowledge about concepts and relationships among them. Then the two knowledge bases are kept in sync by some additional external mechanisms. This is the approach followed, for example, in [45] and, with slight modifications, in [21]. However, in our opinion, it does not apply very well to knowledge domains with more than two levels of nesting, which are not strictly *stratified* (i.e., when an item can be simultaneously member of two or more items placed at different levels). This issue also affects the *Fixed Layered Metamodeling Architecture*, studied in [35, 34].

[3] `http://www.redlist.org`

1.3 The $\forall^\pi$ Family

In this thesis we present a collection of decidable quantified fragments of set theory, called the $\forall^\pi$ family (see Chapter 4), which allow the explicit manipulation of ordered pairs, and which can be profitably employed as knowledge representation languages.

We present a decision procedure for each language of this family, and prove that all of these procedures are optimal (in the sense that they run in nondeterministic polynomial-time) when restricted to formulae with quantifier nesting bounded by a constant. A considerable amount of set-theoretic constructs can be expressed by the languages in this family, also in the restricted case. In particular map-related constructs like map inverse, Boolean operator among maps, map transitivity, and so on, are expressible. In addition, the quantified nature of these languages and the pair-related constructs they provide allow to map numerous description logic constructs into them in a very natural way, showing that these languages can effectively serve to knowledge representation.

A restricted set of SWRL rules, namely those which do not contains *data literals*, can be easily embedded in all the languages of the $\forall^\pi$ family without disrupting decidability.

Finally, as language terms are interpreted as sets in the *von Neumann standard cumulative hierarchy* $\mathcal{V}$, described later, the semantics of our languages is *multi-level*, so that these languages can embody metamodeling in a very natural way.

1.4 Contributions

The main contributions of this thesis are in the field of Computable Set Theory. The decidability of the languages in the $\forall^\pi$ family is helpful in defining the boundary between decidable and undecidable quantified fragments of set theory.

In addition, we consider the proof of the NP-completeness of the satisfiability problem for formulae in the unquantified language $\mathsf{MLSS}^{\times}_{2,m}$, carried out in Section 5.1 by means of a straightforward reduction to the satisfiability problem for formulae in a language of the $\forall^\pi$ family, a significant advancement in the study of set-theoretical languages with map-related operators, in particular languages which allows the Cartesian product. Concerning this, we recall that extending MLS with Cartesian product and cardinality comparison leads to undecidability.

Comparison of Computable Set Theory with the description logics framework brought some interesting results in both these areas. Regarding Computable Set Theory, in Section 5.1.3 the ExpTime-hardness of the unsatisfiability problem for every MLS extension which contains the map image operator is derived from an analogous result for the description logic $\mathcal{ALC}$.

On the description logic side, the result obtained in Section 5.2 shows that imposing some restrictions on the usage of existential quantifier constructs allows one to identify a very expressive description logic, which we denote as $\mathcal{DL}\langle\forall^\pi\rangle$, for which the consistency problem is NP-complete. In addition, it can be equipped with metamodeling-features in such a way that the consistency problem remains NP-complete. Finally, both $\mathcal{DL}\langle\forall^\pi\rangle$ and its metamodeling-enabled version remain decidables also if extended with SWRL rules.

1.5 Thesis Organization

The rest of this thesis is organized as follows. Chapters 2 and 3 provide a short introduction to Computable Set Theory and description logics, respectively, where their basic notions and some languages of our interest are briefly reviewed.

Chapter 4 introduces the three quantified fragments of set theory $\forall^\pi_{\mathbf{0}}$, $\forall^\pi_{\mathbf{0,2}}$, and $\forall^\pi_{\mathbf{\Delta}}$, which together constitute the $\forall^\pi$ family. In the same chapter, these language are proved to have a decidable decision problem, which is NP-complete when restricting to formulae whose quantifier prefix lengths are bounded by a constant.

In Chapter 5 we study the expressive power of $\forall^\pi$ languages. To this purpose, we present the unquantified fragment of set theory $\mathsf{MLSS}^\times_{2,m}$ and the description logic $\mathcal{DL}\langle\forall^\pi\rangle$, which are both expressible in a restriction of $\forall^\pi_{\mathbf{0,2}}$ characterized by an NP-complete decision problem. In the same chapter we show that extending $\mathsf{MLSS}^\times_{2,m}$ (and thus also $\forall^\pi_{\mathbf{0,2}}$) with the map image operator, or with the map domain operator, would trigger the ExpTime-hardness of the decision problem. Finally, we prove that $\mathcal{DL}\langle\forall^\pi\rangle$ can be extended with SWRL rules and metamodeling features, but still remaining decidable.

Finally, in Chapter 6 we draw our conclusions giving also some hints to future works.

Chapter 2

Computable Set Theory

In this chapter we give a brief overview of Computable Set Theory. In particular, we will review MLS, which has been the first sublanguage of set theory to be studied in this context. Then, we will focus our attention on MLS extensions which involve constructs related to multi-valued maps. Finally, the quantified language $\forall_{\mathbf{0}}$ will be discussed. All of the languages mentioned in this chapter have been proved to be decidable. However, no decision procedure is reported here, rather the decidability of some of these fragments will be proved in Chapter 4, by way of reductions to languages in the $\forall^{\pi}$ family, presented there.

To begin with, we define a general setup for set-theoretical languages by introducing some useful notations and definitions.

2.1 Preliminaries

Computable Set Theory focuses on decidable fragments of set theory. We will refer to the Zermelo-Fraenkel axiomatization of set theory, and we will restrict out attention to *pure* sets, i.e. sets whose members are sets recursively founded by the empty set $\emptyset$. We recall that the *von Neumann standard cumulative hierarchy of sets* $\mathcal{V}$ is the class containing all the pure sets. This hierarchy is defined by

$$\begin{array}{rcll} \mathcal{V}_0 & = & \emptyset & \\ \mathcal{V}_{\gamma+1} & = & \mathcal{P}(\mathcal{V}_\gamma)\,, & \text{for each ordinal } \gamma \\ \mathcal{V}_\lambda & = & \bigcup_{\mu<\lambda} \mathcal{V}_\mu\,, & \text{for each limit ordinal } \lambda \\ \mathcal{V} & = & \bigcup_{\gamma\in On} \mathcal{V}_\gamma\,, & \end{array}$$

where $\mathcal{P}(\cdot)$ is the powerset operator and On denotes the class of all ordinals. It can be proved that the membership relation is well-founded in $\mathcal{V}$ and, therefore, no membership cycle can occur in $\mathcal{V}$.

We denote with $\mathsf{rank}(u)$ the least ordinal γ such that $u \subseteq \mathcal{V}_\gamma$ (i.e., $u \in \mathcal{V}_{\gamma+1}$), for every set u in $\mathcal{V}$.

The basic constituents of the languages studied in the context of Computable Set Theory are *variables*. In the rest of this thesis, we will denote

with $Vars = \{x, y, z, \ldots\}$ the denumerable infinite set of variables. In addition, some fragments which allow map-related or pair-related constructs will make also use of another sort of variables, *map variables*, which are intended to denote multi-valued maps; their collection will be denoted by $Vars_m = \{f, g, h, \ldots\}$. For the sake of conciseness, we will assume that $Vars_m$ is just a sub-sort of $Vars$ (i.e. $Vars_m \subseteq Vars$), and we will denote with $Vars_s$ the remaining variables (i.e. $Vars_s = Vars \setminus Vars_m$), which we call *set variables*.

Given any set theoretic formula φ, we will denote with φ_y^x the formula obtained by replacing each occurrence of x in φ with y, with $x, y \in Vars$.

Variables can be combined together to form complex terms by means of the usual set theoretic operators (see Table 2.1, left column). In particular, any term of the form $[X, Y]$, where X, Y are set theoretical terms, represent the ordered pair of X and Y.

Finally, formulae of our languages are built from the *atomic formulae*, described in Table 2.2, using the connectives of propositional logic and, in quantified fragments, also the quantifiers $\forall$ and $\exists$.

In the next sections we provide a *background* semantics, which will be specialized for each fragment of set theory presented in the following. To this purpose it is useful to recall first the notion of ordered pairs.

2.1.1 Ordered Pairs

We remark that multi-valued maps are just sets consisting only of *ordered pairs*. Thus, for our purposes, it is useful to recall some notions concerning ordered pairs in set theory.

Ordered pairs are represented in set theory by means of *pairing functions*. In order to provide the precise definition of such kind of operations, we need first to recall that, given an injective binary operation over sets π, and two sets u, v, the *Cartesian product* of u and v with respect to π, denoted by $u \times v$, can be defined as follows:

$$u \times v = \{\pi(u', v') : u' \in u \wedge v' \in v\}.$$

Definition 1. Let π be a binary operation over sets in $\mathcal{V}$. π is said to be a *pairing function* if and only if the following conditions hold:

(P1) $\pi(u, v) = \pi(u', v') \iff u = u' \wedge v = v'$, and

(P2) $u \times v$ is a set in $\mathcal{V}$

for any $u, v, u', v' \in \mathcal{V}$.

Let π be a pairing function, and let u be a set in the von Neumann hierarchy. We denote with $\pi(u)$ the set consisting of the pair members of u, with respect to the pairing function π. More formally

$$\pi(u) = \{\pi(v, v') : \pi(v, v') \in u\}.$$

Note that from this definition it follows that $\pi(u)$ is a set in $\mathcal{V}$ as u is.

Syntax	$\mathbf{I}(\cdot)$
x (variable)	$M_{\mathbf{I}}x$
$\emptyset$ (empty set)	$\emptyset$
$X \cup Y$ (union)	$\mathbf{I}X \cup \mathbf{I}Y$
$X \cap Y$ (intersection)	$\mathbf{I}X \cap \mathbf{I}Y$
$X \setminus Y$ (difference)	$\mathbf{I}X \setminus \mathbf{I}Y$
$\{X\}$ (singleton)	$\{\mathbf{I}X\}$
$[X, Y]$ (pair)	$\pi_{\mathbf{I}}(\mathbf{I}X, \mathbf{I}Y)$
X^{-1} (map inverse)	$\{\pi_{\mathbf{I}}(u, v) : \pi_{\mathbf{I}}(v, u) \in \mathbf{I}X\}$
$\mathsf{dom}(X)$ (domain)	$\{u : (\exists v)(\pi_{\mathbf{I}}(u, v) \in \mathbf{I}X\}$
$\mathsf{range}(X)$ (range)	$\mathbf{I}(\mathsf{dom}(X^{-1}))$
$X[Y]$ (image)	$\{v : (\exists u)(\pi_{\mathbf{I}}(u, v) \in \mathbf{I}X \wedge u \in \mathbf{I}Y)\}$
$X_{Y\|}$ (domain restriction)	$\{\pi_{\mathbf{I}}(u, v) : \pi_{\mathbf{I}}(u, v) \in \mathbf{I}X \wedge u \in \mathbf{I}X\}$
$X_{\|Y}$ (range restriction)	$\{\pi_{\mathbf{I}}(u, v) : \pi_{\mathbf{I}}(u, v) \in \mathbf{I}X \wedge v \in \mathbf{I}X\}$
$X_{Y\|Z}$ (map restriction)	$\mathbf{I}(X_{Y\|} \cap X_{\|Z})$
$X \circ Y$ (composition)	$\{\pi_{\mathbf{I}}(u, v) : (\exists u', v')(\pi_{\mathbf{I}}(u, u') \in \mathbf{I}X \wedge$ $\pi_{\mathbf{I}}(v', v) \in \mathbf{I}Y)\}$
$\mathsf{id}(X)$ (identity)	$\{\pi_{\mathbf{I}}(u, u) : u \in \mathbf{I}X\}$
$\mathsf{sym}(X)$ (symmetric closure)	$\mathbf{I}(X \cup \mathbf{I}X^{-1})$
$\mathsf{refl}(X)$ (reflexive closure)	$\mathbf{I}(X \cup \mathsf{id}(\mathsf{dom}(X)))$
X^+ (transitive closure)	$\{\pi_{\mathbf{I}}(u, v) : \pi_{\mathbf{I}}(u, v) \in \mathbf{I}X \vee (\exists v')(\pi_{\mathbf{I}}(u, v') \in \mathbf{I}X \wedge$ $\mathbf{I}(v', v) \in \mathbf{I}X\}$
X^* (transitive reflexive closure)	$\mathbf{I}(\mathsf{refl}(X) \cup X^+)$

Table 2.1: *Set-theoretic terms*

Syntax	$\mathbf{I} \models (\cdot)$ iff
$X \in Y$	$\mathbf{I}X \in \mathbf{I}Y$
$X = Y$	$\mathbf{I}X = \mathbf{I}Y$
$X \subseteq Y$	$\mathbf{I}X \subseteq \mathbf{I}Y$
$\mathsf{injective}(F)$	$(\forall u, u', v)(\pi_{\mathbf{I}}(u, v) \in \mathbf{I}F \wedge \pi_{\mathbf{I}}(u', v) \in \mathbf{I}F \rightarrow u = u')$
$\mathsf{single\_valued}(F)$	$\mathbf{I} \models \mathsf{injective}(F^{-1})$
$\mathsf{bijective}(F)$	$\mathbf{I} \models \mathsf{injective}(F) \wedge \mathbf{I} \models \mathsf{single\_valued}(F)$
$\mathsf{is\_reflexive}(F)$	$\mathbf{I}(F^{-1}) \subseteq \mathbf{I}F$
$\mathsf{is\_transitive}(F)$	$\mathbf{I}(F \circ F) \subseteq \mathbf{I}F$
$\mathsf{is\_asym}(F)$	$\mathbf{I}F \cap \mathbf{I}(F^{-1}) = \emptyset$
$\mathsf{is\_irreflexive}(F)$	$\mathbf{I}F \cap \mathbf{I}(\mathsf{refl}(F)) = \emptyset$

Table 2.2: *Set-theoretic atomic formulae*

2.1.2 Background Semantics

Now we provide some notions and definitions that will be used in the semantics definitions of the fragments presented later.

To begin with, we will refer to mappings from variables to sets in $\mathcal{V}$ as *assignments*. Let W be a finite subset of *Vars*, and let M, M' be assignments. Then M' is said to be a *W-variant* of M if $M'x = Mx$ for all $x \in \mathit{Vars} \setminus W$ (i.e. M and M' coincides except for the variables in W).

A *set-theoretic interpretation* is a pair $\mathbf{I} = (M_{\mathbf{I}}, \pi_{\mathbf{I}})$, where $M_{\mathbf{I}}$ is an assignment, and $\pi_{\mathbf{I}}$ is a *pairing function*. We characterize this kind of interpretation as *set-theoretic* in order to distinguish them from the *descriptive* interpretations of description logic, which will be defined in Chapter 3. However, we will omit to indicate the interpretation type when it will be clear from the context.

The notion of W-variant is extended to interpretations as expected. Thus, let W be a finite subset of *Vars*, and let $\mathbf{I}$, $\mathbf{I}'$ be two interpretations. $\mathbf{I}'$ is said to be a W-variant of $\mathbf{I}$ if and only if

- $\pi_{\mathbf{I}'} = \pi_{\mathbf{I}}$, and
- $M_{\mathbf{I}'}$ is a W-variant of $M_{\mathbf{I}}$.

An interpretation $\mathbf{I}$ associates sets in $\mathcal{V}$ to terms as indicated in Table 2.1, right column. Furthermore, interpretations *evaluate* set-theoretic formulae to a truth value **true** or **false**. We say that an interpretation $\mathbf{I}$ is a *model* for a formula φ if $\mathbf{I}$ evaluates φ to **true**. In this case, we write $\mathbf{I} \models \varphi$. Otherwise, if $\mathbf{I}$ evaluates a formula φ to **false**, we write $\mathbf{I} \not\models \varphi$.

A formula is said to be *satisfiable* if it admits a model, otherwise it is said to be *unsatisfiable*. Thus, the *satisfiability problem* (in short s.p.) consists in determining whether a formula is satisfiable or not.

Evaluation of formulae for each of the languages described in this chapter will be defined in a precise way when the languages will be presented.

However, evaluation of atomic formulae is carried out as indicated in Table 2.2, right column.

We conclude this section with some definitions concerning interpretations. Let W be any collection of variables. An interpretation $\mathbf{I}$ is said to be *pair-free* with respect to W if

$$\pi_{\mathbf{I}}(M_{\mathbf{I}}x) = \emptyset$$

for each $x \in W$, that is, $\mathbf{I}$ associates to variables in W only sets which do not contain any pair. An interpretation is said to be just pair-free if it is pair-free with respect to the whole set of variables *Vars*.

Finally, an interpretation $\mathbf{I}$ is said to be *pair-safe* if it associates just sets of ordered pairs (or the empty set) to map variables. In other words, $\mathbf{I}$ is pair-safe if and only if

$$M_{\mathbf{I}}f = \pi_{\mathbf{I}}(M_{\mathbf{I}}f)$$

for each $f \in Vars_m$.

In the next section we will examine some unquantified fragments of set theory, starting from the basic languages MLS and MLSS, and then reviewing some MLSS extensions with map constructs.

2.2 Multi-Level Syllogistic

As mentioned before, *Multi-Level Syllogistic* (in short MLS) has been the first unquantified language studied in the context of Computable Set Theory. MLS is a fragment of set theory which contains:

1. the denumerable infinity of variables *Vars*;
2. the set-theoretical operators $\cup, \cap, \setminus$;
3. the relators $\in, =$;
4. the Boolean connectives of propositional logic.

The set of the terms allowed in this language, which we call the set of MLS-terms, is defined by the following syntax rule:

$$X, Y \longrightarrow x \mid X \cup Y \mid X \cap Y \mid X \setminus Y$$

where x is a variable, and X, Y are MLS-terms. Atomic MLS-formulae are atomic formulae of the types

$$X \in Y, \quad X = Y$$

where X and Y are MLS-terms. Finally, MLS-formulae are Boolean combinations of atomic MLS-formulae by means of the connectives of propositional logic. Then, evaluation of MLS-formulae by interpretations is carried out as

usual in propositional logic, assuming that atomic formulae are evaluated as indicated in Table 2.2.

The language MLSS (Multi-Level Syllogistic with Singleton) is obtained by extending MLS with the singleton operator $\{X\}$. In more details, the set of MLSS *-terms* is defined by

$$X, Y \longrightarrow x \mid X \cup Y \mid X \cap Y \mid X \setminus Y \mid \{X\}$$

where x is a variable, and X, Y are MLSS-terms. Again, MLSS *-formulae* are Boolean combinations of atomic MLSS-formulae, and evaluation of MLSS-formulae follows the standard rules.

In the next section we review some extensions involving map constructs. It must be noticed that the satisfiability problem for formulae of the languages presented in the next section is ExpTime-hard, as reported in Theorem 2.

2.2.1 Multi-Level Syllogistic with Maps

The first MLS extension involving maps we examine is that studied in [17], which we denote by $\mathsf{MLS_m}$, where the subscript indicates the presence of map related operators. $\mathsf{MLS_m}$ *-terms* are formed according to the following syntax rule:

$$\begin{array}{rcl} X, Y, Z & \longrightarrow & x \mid X \cup Y \mid X \cap Y \mid X \setminus Y \mid \mathsf{dom}(X) \mid \mathsf{range}(X) \mid \\ & & X[Y] \mid X^{-1} \mid X_{Y|} \mid X_{|Y} \mid X_{Y|Z} \end{array}$$

where $x \in \mathit{Vars}$, and X, Y, Z are $\mathsf{MLS_m}$-terms. *Atomic* $\mathsf{MLS_m}$ *-formulae* are of the following types:

$$\begin{array}{lll} [X, Y] \in Z, & X \in Y, & X = Y, \\ \mathsf{injective}(X), & \mathsf{single\_valued}(X), & \mathsf{bijective}(X), \end{array}$$

where X, Y, Z are $\mathsf{MLS_m}$-terms. Finally, $\mathsf{MLS_m}$-*formulae* are Boolean combinations of atomic MLSS-formulae.

Another approach to extend MLS with map constructs is the one followed in [19], where a two-sorted language which contains map constructs was proved to be decidable. We denote this language as $\mathsf{MLS_{m,2}}$, where the subscript "2" indicates that it contains two sort of variables.

$\mathsf{MLS_{m,2}}$ is a fragment of set theory which contains:

1. the denumerable infinity of set variables Vars_s;
2. the denumerable infinity of map variables Vars_m;
3. the predicate symbols single_valued, bijective;
4. the operator symbols $\cup$, $\cap$, $\setminus$, dom, range, f^{-1} (map inverse), $f[x]$ (image);

5. the relators $\in, =$;

6. the Boolean connectives of propositional logic.

$\mathsf{MLS}_{\mathsf{m},2}$ *-terms* are defined according to the following syntax rules:

$$X, Y \longrightarrow x \mid X \cup Y \mid X \cap Y \mid X \setminus Y \mid f \mid \mathsf{dom}(F) \mid \mathsf{range}(F) \mid f[X] \mid f^{-1}[X]$$

where x and f stay respectively for set and map variables, and X, Y for $\mathsf{MLS}_{\mathsf{m},2}$-terms. *Atomic* $\mathsf{MLS}_{\mathsf{m},2}$ *-formulae* are set-theoretic expressions of the types $X \in Y$, $X = Y$, $\mathsf{single\_valued}(f)$, and $\mathsf{bijective}(f)$, where X, Y are $\mathsf{MLS}_{\mathsf{m},2}$-terms, and f is a map variable. We remark that the decidability of the s.p. for $\mathsf{MLS}_{\mathsf{m},2}$-formulae was proved in [19] with presenting a double exponential-time decision procedure.

Up to this point, we illustrated just unquantified languages devised in the context of Computable Set Theory. In the next section we review a quantified fragment, called $\forall_{\mathbf{0}}$.

2.3 The Language $\forall_{\mathbf{0}}$

In this section we discuss the quantified fragment of set theory $\forall_{\mathbf{0}}$, studied in [6]. We recall that decision procedures for this language can be found in [6, 8]. In particular, the decision procedure presented in [8] is optimal, since it runs in nondeterministic-polynomial time. However, we do not report this decision procedure, as we provide instead a reduction of the satisfiability problem for formulae of this language to one of the languages in the $\forall^{\pi}$-family (which will be proved to be decidable later), that will be introduced and discussed in Chapter 4.

The language $\forall_{\mathbf{0}}$ contains:

- the *variables* in *Vars*;
- the set relators $\in, =$;
- the universal quantifiers $\forall, \exists$;
- parentheses;
- the logical connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$.

The set of the $\forall_{\mathbf{0}}$ *-formulae* is defined as follows. *Atomic* $\forall_{\mathbf{0}}$ *-formulae* are atomic formulae of the types $x \in y, x = y$ (pairs are not allowed in this language), with $x, y \in$ *Vars*. *Quantifier-free* $\forall_{\mathbf{0}}$ *-formulae* are Boolean combinations of atomic $\forall_{\mathbf{0}}$-formulae by means of the propositional connectives. Finally, $\forall_{\mathbf{0}}$-formulae[1] are quantified formulae of the form

$$Q_1 Q_2 \dots Q_n \psi$$

[1] In [6] they are called *simple prenex formulae*.

where

- ψ is a quantifier-free $\forall_{\mathbf{0}}$-formula;
- n is a non-negative integer;
- for $1 \leq i \leq n$, either every Q_i is a restricted quantifier of the form $(\forall x_i \in y_i)$, or every Q_i is of the form $(\exists x_i \in y_i)$ (we will refer to x_i as *quantified variables* and to y_i as *domain variables*, for $1 \leq i \leq n$);
- nesting among quantified variables is not allowed (more precisely, no y_j is a x_i, for all $1 \leq i, j \leq n$).

It must be noticed that, in the previous definition, if we restrict to the case $n = 0$ we get the class of quantifier-free $\forall_{\mathbf{0}}$-formulae. In other words, all the quantifier-free $\forall_{\mathbf{0}}$-formulae are $\forall_{\mathbf{0}}$-formulae.

Given any $\forall_{\mathbf{0}}$-formula φ, we denote with $Vars(\varphi)$ the set of the *free* variables in φ, i.e. all the variables which occur not bounded by any quantifier in φ.

The $\forall_{\mathbf{0}}$ semantics extends the background semantics introduced above to cope with universal quantifiers. Then, evaluation of quantifier-free formulae is carried out as usual, while, given any interpretation $\mathbf{I}$, $\mathbf{I}$ evaluates to **true** a quantified $\forall_{\mathbf{0}}$-formulae of the form $(\forall x \in y)\chi$, with $x, y \in Vars$, if and only if $\mathbf{I}' \models \chi$, for every $\{x\}$-variant $\mathbf{I}'$ of $\mathbf{I}$ such that $\mathbf{I}'x \in \mathbf{I}y$.

Chapter 3

Description Logics

Description Logics are a well-established and well-studied framework for knowledge representation (see [2] for a quite complete introduction to this family of languages). They are logic-based languages which allow to provide a *high-level* description of the world in a formal and precise way. More in details, knowledge about a domain of interest is described in terms of *concepts*, which denote sets of elements, *roles*, which represent relations between elements, and *individuals*, that denote domain elements. Then, a description logic *knowledge base* is a finite description of the world, expressed by means of a finite set of constraints involving concepts, roles, and individuals.

Over the years, description logics have been employed in several application domains such as natural language processing, configuration, data bases and medicine. Most notably, they have been adopted in the context of Semantic Web (cf. [5]) to provide solid semantical foundations to the Semantic Web Language OWL[1] (see for example [25]).

In contrast with other earlier knowledge representation approaches such as, for example, *Semantic Networks* (cf. [40]), description logics are equipped with a well-defined syntax, and a formal, unambiguous semantics. In addition, the description logic framework provides effective mechanisms for *reasoning*, which allow to extract implicit knowledge from the knowledge explicitly stated in a knowledge base. The main *reasoning tasks* identified in this context are:

- *concept satisfiability*, which allows to determine whether a complex concept can be *interpreted* (description logic interpretation will be defined in a precise way later in this chapter) by a non-empty set of domain elements;
- *concepts subsumption*, i.e. testing if a concept *subsumes* another concept;
- *instance checking*, which allows to determine whether an individual belongs to a given concept;

[1] `http://www.w3.org/standards/techs/owl`

- *instance retrieval*, which consists in retrieving all the individuals belonging to a given concept;
- *entailment*, i.e. testing if, given two knowledge bases $\mathcal{K}$ and $\mathcal{K}'$, the facts stated in $\mathcal{K}'$ are consequences of those contained in $\mathcal{K}$.

However, the main reasoning task for description logics is *consistency check*, which consists in determining whether the information contained in a knowledge base is self-contradictory or not, since, except for some weaker description logics, all the other reasoning tasks can be reduced to it. For this reason, when we will use the term "reasoning" without being more specific, we will refer to consistency check. Consistency check, along with the other reasoning tasks, will be defined formally in Section 3.1.

In the last decades sound and complete algorithms were devised for a considerable amount of description logics, and optimized versions of these algorithms have been implemented in several reasoning engines like *FaCT*, *Race*, and *DLP* (cf. [23, 22, 39]).

The trade-off between the expressive power (in terms of allowed constructs) and the computational complexity of reasoning is actually one of the major research topics in description logics. In fact, while the reasoning for the basic description logic $\mathcal{AL}$ is ExpTime-hard (cf. [2, Theorem 3.27, page 132]), and this may be prohibitive for practical applications, a lower computational complexity can be achieved by choosing appropriately the set of allowed constructs (see [7] and [1] for some examples of description logics with polynomial-time reasoning).

This chapter aims to provide a brief overview of the description logic framework, by reviewing some of these logics of our interest. We introduce this framework by describing the basic description logic $\mathcal{ALC}$, and then we discuss some of its *expressive* extensions. Finally, we will examine an extension of this framework based on Horn-like rules, namely SWRL rules, proposed in [26]. As we did in Chapter 2, we begin with providing the general notions and definitions of this framework.

3.1 Preliminaries

As mentioned before, a description logic knowledge base *describes* a knowledge domain in terms of concepts, roles, and individuals. The *building blocks* of description logics are

- a countably infinite collection of *concept names* $\mathcal{N}^c = \{A, B, \ldots\}$,
- a countably infinite collection of *role names* $\mathcal{N}^r = \{P, Q, \ldots\}$, and
- a countably infinite collection of *individual names* $\mathcal{N}^i = \{a, b, \ldots\}$.

Each description logic is mainly characterized by a set of concept and role *constructors*, which, starting from the basic names in $\mathcal{N}^c$, $\mathcal{N}^r$, and $\mathcal{N}^i$,

Constructor	*Interpretation* $(\cdot)^{\mathcal{I}}$
$\top$ (universal concept)	$\Delta^{\mathcal{I}}$
$\bot$ (bottom concept)	$\emptyset$
$\neg C$ (concept negation)	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$C \sqcup D$ (concept union)	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$C \sqcap D$ (concept intersection)	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$\{a\}$ (nominal)	$\{a^{\mathcal{I}}\}$
$\exists R.\texttt{Self}$ (self restriction)	$\{u \in \Delta^{\mathcal{I}} : [u,u] \in R^{\mathcal{I}}\}$
$\forall R.C$ (value restriction)	$\{u \in \Delta^{\mathcal{I}} : (\forall [u,v] \in R^{\mathcal{I}})(v \in C^{\mathcal{I}})\}$
$\exists R.C$ (existential quantifier)	$\{u \in \Delta^{\mathcal{I}} : (\exists v \in C^{\mathcal{I}})([u,v] \in R^{\mathcal{I}})\}$
$\leq nR.C$ (min-cardinality restriction)	$\{u \in \Delta^{\mathcal{I}} : \lvert\{v \in C^{\mathcal{I}} : [u,v] \in R^{\mathcal{I}}\}\rvert \leq n\}$
$\geq nR.C$ (max-cardinality restriction)	$\{u \in \Delta^{\mathcal{I}} : \lvert\{v \in C^{\mathcal{I}} : [u,v] \in R^{\mathcal{I}}\}\rvert \geq n\}$

Table 3.1: *Description logic* concept *constructors*

allow to form complex concepts and roles. Thus, description logic terms are divided into concept and role terms: *concept terms* are concept names and complex concepts constructed by means of the concept constructors; analogously, *role terms* are role names and complex roles constructed by means of role constructors. Some of the most common constructors used in description logics are listed in Tables 3.1 and 3.2, where C, D are concepts, R, S are roles, and a is an individual name.

In general, we will use the terms "concepts" and "roles" to indicate concept and role terms, respectively.

Description logics are also characterized by the types of constraints they allow to be specified in knowledge bases. These constraints (see Table 3.3) can be of three different types: concepts inclusions of the form $C \sqsubseteq D$, role inclusions $R \sqsubseteq S$, and assertions like $C(a)$, $R(a,b)$, concerning the membership of individuals or individual pairs to concepts and roles, respectively. In addition, some expressive description logic allow also constraints stating properties of roles, for example $\texttt{Trans}(R)$, which expresses the fact that the

Constructor	Interpretation $(\cdot)^{\mathcal{I}}$
$\mathbb{U}$ (universal role)	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
$\neg R$ (role negation)	$(\Delta \times \Delta) \setminus R^{\mathcal{I}}$
$R \sqcup S$ (role union)	$R^{\mathcal{I}} \cup S^{\mathcal{I}}$
$R \sqcap S$ (role intersection)	$R^{\mathcal{I}} \cap S^{\mathcal{I}}$
R^{-} (role inverse)	$\{[u, v] \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} : [v, u] \in R^{\mathcal{I}}\}$
$R_{C\|}$ (role domain restriction)	$\{[u, v] \in R^{\mathcal{I}} : u \in C^{\mathcal{I}}\}$
$R_{\|D}$) (role range restriction)	$\{[u, v] \in R^{\mathcal{I}} : v \in D^{\mathcal{I}}\}$
$R_{C\|D}$ (role domain and range restriction)	$(R_{C\|})^{\mathcal{I}} \cap (R_{\|D})^{\mathcal{I}}$
$\mathtt{id}(C)$ (role identity)	$\{[u, u] : u \in C^{\mathcal{I}}\}$
$R \circ S$ (role composition)	$R^{\mathcal{I}} \circ S^{\mathcal{I}}$
R^{*} (transitive closure)	$(R^{\mathcal{I}})^{*}$
$\mathtt{sym}(R)$ (symmetric closure)	$R^{\mathcal{I}} \cup (R^{-})^{\mathcal{I}}$

Table 3.2: *Description logic* role *constructors*

Syntax	$\mathcal{I} \models (\cdot)$ iff
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
$R \equiv S$	$R^{\mathcal{I}} = S^{\mathcal{I}}$
$\texttt{Sym}(R)$	$(\forall [u,v] \in R^{\mathcal{I}})([v,u] \in R^{\mathcal{I}})$
$\texttt{Trans}(R)$	$(\forall [u,v],[v,v'] \in R^{\mathcal{I}})([u,v'] \in R^{\mathcal{I}})$
$\texttt{Ref}(R)$	$(\forall [u,v] \in R^{\mathcal{I}})([u,u] \in R^{\mathcal{I}})$
$\texttt{ASym}(R)$	$(\forall [u,v] \in R^{\mathcal{I}})([v,u] \notin R^{\mathcal{I}})$
$\texttt{Irr}(R)$	$(\forall [u,v] \in R^{\mathcal{I}})([u,u] \notin R^{\mathcal{I}})$
$\texttt{Dis}(R,S)$	$R^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset$
$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
$R(a,b)$	$[a^{\mathcal{I}}, b^{\mathcal{I}}] \in R^{\mathcal{I}}$

Table 3.3: *Knowledge base constraints*

role R must be a transitive relation.

Then a description logic *knowledge base* is a finite set of constraints of the types listed in Table 3.3, left column, where C, D are concepts, R, S are roles, and a, b are individual names.

Description logic semantics[2] is given in terms of *descriptive interpretations*. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty set $\Delta^{\mathcal{I}}$, namely the interpretation *domain*, and an interpretation function $\cdot^{\mathcal{I}}$ assigning to each concept name a subset of $\Delta^{\mathcal{I}}$, to every role name a relation over $\Delta^{\mathcal{I}}$, and to every individual name a domain item in $\Delta^{\mathcal{I}}$. An interpretation $\mathcal{I}$ extends recursively to complex terms as indicated in the right column of Tables 3.1 and 3.2.

An interpretation $\mathcal{I}$ evaluates a description logic constraint γ to a truth value **true** or **false**. We say that an interpretation $\mathcal{I}$ *satisfies* a constraint γ, and write $\mathcal{I} \models \gamma$, if $\mathcal{I}$ evaluates γ to **true**, otherwise we write $\mathbf{I} \not\models \gamma$. Evaluation of knowledge base constraints is carried out as indicated in the right column of Table 3.3.

If an interpretation $\mathcal{I}$ satisfies all the constraints in a knowledge base $\mathcal{K}$, then $\mathcal{I}$ is said to be a *model* for $\mathcal{K}$, and we write $\mathcal{I} \models \mathcal{K}$. A knowledge base is said to be *consistent* if it admits a model. Thus the *consistency problem* for description logic knowledge bases is to determine whether a knowledge base is consistent or not.

We conclude this section with the precise definitions of the other reasoning tasks mentioned before.

[2]Here we will limit ourselves to the *descriptive* semantics. There are several other semantics that are out of the scope of this thesis.

Given a concept C, we say that C is *satisfiable* if and only if there exists an interpretation $\mathcal{I}$ which associates to it a non-empty set, i.e. $C^{\mathcal{I}} \neq \emptyset$. Given a knowledge base $\mathcal{K}$ and a concept C, C is said to be *satisfiable with respect to* $\mathcal{K}$ if $\mathcal{K}$ admits a model $\mathcal{I}$ such that $C^{\mathcal{I}}$ is not empty.

Now let C, D be two concepts. We say that C *subsumes* D if and only if $D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$, for every interpretation $\mathcal{I}$. In addition, we say that C subsumes D with respect to the knowledge base $\mathcal{K}$ if and only if $D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ for each model $\mathcal{I}$ of $\mathcal{K}$.

Given a knowledge base $\mathcal{K}$, a concept C, and an individual a, then a is said to be an *instance* of C with respect to $\mathcal{K}$ if and only if $\mathcal{I} \models C(a)$ for every model $\mathcal{I}$ of $\mathcal{K}$.

The reasoning tasks called *instance retrieval* consists in determining which individuals are instances of the concept C with respect to the knowledge base $\mathcal{K}$.

Finally, given two knowledge bases $\mathcal{K}$ and $\mathcal{K}'$, we say that $\mathcal{K}$ *entails* $\mathcal{K}'$ if each model of $\mathcal{K}$ is a model for $\mathcal{K}'$ also, i.e. $\mathcal{I} \models \mathcal{K} \Longrightarrow \mathcal{I} \models \mathcal{K}'$.

In the next section we recall the basic description logic $\mathcal{ALC}$ and then show that in $\mathcal{ALC}$ all the reasoning tasks we just mentioned can be reduced to knowledge base consistency.

3.2 The Description Logic $\mathcal{ALC}$

$\mathcal{ALC}$ is one of the earlier languages devised in the context of description logics. It contains:

- a countably infinite collection of *concept names* $\mathcal{N}^c = \{A, B, \ldots\}$;
- a countably infinite collection of *role names* $\mathcal{N}^r = \{P, Q, \ldots\}$;
- the concept constructors $\sqcup$, $\sqcap$, $\neg$, $\exists$, $\forall$;
- concept inclusions and equality symbols.

The collection of $\mathcal{ALC}$-concepts is defined by the following syntax rule:

$$C, D \longrightarrow A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists P.C \mid \forall P.C$$

where C, D are $\mathcal{ALC}$-concepts, A is a concept name in $\mathcal{N}^c$, and P is a role name, while complex roles are not allowed in $\mathcal{ALC}$. Finally, an $\mathcal{ALC}$-knowledge base is a finite set of constraints of the following types:

$$C \sqsubseteq D, \quad C \equiv D, \quad C(a), \quad R(a, b)$$

with C, D $\mathcal{ALC}$-concepts, and $a, b \in \mathcal{N}^i$.

The description logic $\mathcal{ALC}$ extends the basic description logic $\mathcal{AL}$ (which is an acronym for *Attributive Language*) by also allowing the negation of

complex concepts, while in $\mathcal{AL}$ concept negation is available only for concept names.

This peculiarity of $\mathcal{ALC}$ allows one to reduce all the reasoning tasks mentioned above to knowledge base consistency, since complex negation allows one to express the negation of $\mathcal{ALC}$-knowledge base constraints.

Plainly, a concept C is satisfiable with respect to a knowledge base $\mathcal{K}$ if and only if the knowledge base $\mathcal{K} \cup \{C(a)\}$, where a is an individual name not already occurring in $\mathcal{K}$ and in C, is consistent. In turn, a concept C is satisfiable (in general) if and only if it is satisfiable with respect to the empty knowledge base.

Concept subsumption can be reduced to concept (un)satisfiability, which we just proved to be reducible to knowledge base consistency, since, given any two concepts C and D, C subsumes D if and only if the concept $C \sqcap \neg D$ is unsatisfiable.

Next, we show that the entailment of a knowledge base $\mathcal{K}'$ by $\mathcal{K}$ can be reduced to knowledge base (un)consistency. Let us consider first the simpler case in which $\mathcal{K}'$ consists of a single constraint γ. If γ is the concept inclusion $C \sqsubseteq D$, for some concepts C and D, then $\mathcal{K}$ entails $\mathcal{K}' = \{C \sqsubseteq D\}$ if and only if the knowledge base $\mathcal{K} \cup \{(C \sqcap \neg D)(a)\}$ is not consistent, for some individual name a not already occurring in $\mathcal{K} \cup \mathcal{K}'$. Analogously, $\mathcal{K}$ entails $\{C(b)\}$, where C is a concept and b is an individual name, if and only if $\mathcal{K} \cup \{(\neg C)(b)\}$ is inconsistent.[3]

Now let $\mathcal{K}' = \{\gamma_1, \ldots \gamma_n\}$, for some $\mathcal{ALC}$-knowledge base constraints $\gamma_1, \ldots, \gamma_n$, with $n \in \mathbb{N}, n > 1$. Then $\mathcal{K}$ entails $\mathcal{K}'$ if and only if $\mathcal{K}$ entails γ_i, for every $1 \leq i \leq n$.

Finally, a knowledge base $\mathcal{K}$ entails that the individual a is an instance of the concept C if and only if $\mathcal{K}$ entails the knowledge base $\{C(a)\}$, and all the instances of C with respect to $\mathcal{K}$ can be retrieved by iterating the instance checking test over all the individual names occurring in $\mathcal{K}$.

3.2.1 Complexity Issues

It must be noticed that the description logic $\mathcal{ALC}$ is not very expressive. For example, it does not contains complex roles. Despite of this, the complexity of reasoning in $\mathcal{ALC}$ is ExpTime-hard, and this is prohibitive for some application domains.

This complexity lower bound was proved for the weaker description logic $\mathcal{AL}$ in [2, Theorem 3.27, page 132], where the accessibility problem for *succinct* representations of AND-OR graphs was reduced to concept unsatisfiability with respect to knowledge base consisting of concept inclusions only (see [36, 3]). Hence, the following Theorem easily follows, since $\mathcal{AL}$ is a sublogic of $\mathcal{ALC}$.

[3]Entailment of role assertions of the form $R(a, b)$ requires an additional constructor, called *nominal*.

Theorem 2. *The problem of deciding whether a given $\mathcal{ALC}$-concept C is unsatisfiable with respect to a finite collection of $\mathcal{ALC}$-inclusions is* ExpTime*-hard.* □

We argue that this lower bound is mainly due to the presence of the quantifiers $\exists$ and $\forall$ in an unconstrained way, since their presence force any decision algorithm to *invent* new items when constructing a model witnessing the consistency of a knowledge base. Intuitively speaking, if during the construction process of such a model we find an item $u \in (\exists R.C)^{\mathcal{I}}$, we may be forced to introduce a new item $v \in C^{\mathcal{I}}$ to be placed as second item of the pair $[u, v] \in R^{\mathcal{I}}$. And, in $\mathcal{ALC}$, this may require to introduce exponentially many new items.

This argumentation is supported by the fact that reasoning in the description logic $\mathcal{DL}\langle\forall^{\pi}\rangle$ (studied in Section 5.2) is NP-complete despite of the significant amount of constructs available in this language, since the usage of quantifiers is restricted in this description logic.

In the next sections we will examine some expressive extensions of $\mathcal{ALC}$.

3.3 Number Restrictions

In general, the expressive power of $\mathcal{ALC}$ is considered insufficient to deal with knowledge representation matters. For this reason, over the years several extensions of $\mathcal{ALC}$ have been studied. In this context the so called *number description* were devised. These are actually considered one of the most distinguishing feature of the description logic framework. Number descriptions, described in Table 3.1, are concept constructors which allow to impose some cardinality constraints. For example, the concept $\leq 3R.C$, where R is a role and C is a concept, contains all those domain items which are connected by the role R to at most 3 domain items in the concept C. Thus, for example, the following constraint states that the role R is single-valued[4]

$$\top \sqsubseteq \leq 1R.\top.$$

Observe that the constructors $\leq n$ and $\geq n$ are mutually expressible in those description logics which allow complex concepts negation, since $\leq nR.C \equiv \neg(\geq (n+1)R.C)$ and $\geq nR.C \equiv \neg(\leq (n+1)R.C)$ are valid statements.

In addition, an existential restriction $\exists R.C$ can be expressed as $\geq 1R.C$, so that the complexity upper bound provided by Theorem 2 holds also for the description logic obtained from $\mathcal{ALC}$ extended with number restriction, but disallowing the quantifiers.

[4]Here we used the term "single-valued", which is typical of set-theoretical languages, just to avoid confusion. However, in the description logic context these kind of roles are often called *functional* roles.

In the next section we present the very expressive description logic $\mathcal{SROIQ}$, which underpins the semantic web language OWL2.

3.4 The Description Logic $\mathcal{SROIQ}$

The description logic $\mathcal{SROIQ}$, devised in [24], is a very powerful description logic which extends $\mathcal{ALC}$ with some advanced features concerning roles. In fact, it allows role inverses, the universal role, role inclusion constraints and role composition. However, it imposes some quite intricate syntactic restrictions. Nevertheless, the consistency problem in this language is NExpTime-hard, since it extends the description logic $\mathcal{ALCQIO}$, whose NExpTime-hardness was proved in [44].

$\mathcal{SROIQ}$ concepts and roles are defined by means of the following syntax rules:

$$\begin{array}{rcl} C, D & \longrightarrow & A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C \mid \{a\} \mid \exists C.\texttt{Self} \mid \\ & & \leq nR.C \mid \ \geq nR.C \\ R & \longrightarrow & P \mid R^- \end{array}$$

where n is a non-negative integer, A is a concept name, C and D are $\mathcal{SROIQ}$-concepts, a is an individual name, P is a role name, and R is a $\mathcal{SROIQ}$-role. Semantics of $\mathcal{SROIQ}$ concepts and roles is reported in the right column of Tables 3.1 and 3.2, respectively.

Beside the constraints allowed in $\mathcal{ALC}$-knowledge bases, a $\mathcal{SROIQ}$-knowledge base may contain also role inclusions, and role assertions of the following types:

$$\begin{array}{lll} \texttt{Sym}(R), & \texttt{ASym}(R), & \texttt{Trans}(R), \\ \texttt{Ref}(R), & \texttt{Irr}(R), & \texttt{Dis}(R, S), \end{array}$$

where R and S are $\mathcal{SROIQ}$-roles. However, as mentioned before, $\mathcal{SROIQ}$ imposes several syntactic restrictions on knowledge base, which require some preliminary definition to be stated.

Let $\prec$ be a *strict partial order* over the set of $\mathcal{SROIQ}$-roles (we recall that a relation is said to be a strict partial order if it is irreflexive and transitive). Then $\prec$ is said to be *regular* if the following holds for any pair of $\mathcal{SROIQ}$-roles R and S:

$$S \prec R \iff S^- \prec R.$$

A *role hierarchy* $\mathcal{R}_h$ is a finite set of inclusions of the form

$$R_1 \circ \ldots \circ R_n \sqsubseteq P$$

where $R_1, \ldots, R_n$ are $\mathcal{SROIQ}$-roles not including the universal role $\mathbb{U}$, and P is a role name.

Now let $\prec$ be a strict partial order, and let $w \sqsubseteq P$ be a role inclusion, with $w = R_1 \circ \ldots \circ R_n$ for some $n > 0$ and $R_1, \ldots, R_n$ $\mathcal{SROIQ}$-roles different from $\mathbb{U}$. Then $w \sqsubseteq P$ is said to be $\prec$*-regular* if either one of the following conditions holds, for some role name P and for some roles $S_1, \ldots, S_{n-1}$:

- $w = P \circ P$;
- $w = P \circ P^{-}$;
- $R_i \prec P$, for all $1 \leq i \leq n$;
- $w = P \circ S_1 \circ \ldots \circ S_{n-1}$ and $S_i \prec P$, for all $1 \leq i < n$;
- $w = S_1 \circ \ldots \circ S_{n-1} \circ P$ and $S_i \prec P$, for all $1 \leq i < n$.

Finally, a role hierarchy $\mathcal{R}_h$ is said to be *regular* if there exists a regular order $\prec$ such that each inclusion in $\mathcal{R}_h$ is $\prec$-regular.

Given a role hierarchy $\mathcal{R}_h$, we denote with $\sqsubseteq^*$ the reflexive transitive closure of the relation $\sqsubseteq$ among roles induced by $\mathcal{R}_h$. Then, a $\mathcal{SROIQ}$-role R is said to be a subrole of the $\mathcal{SROIQ}$-role S if $R \sqsubseteq^* S$.

Role Assertions are knowledge base constraints of the following forms:

$$\begin{array}{lll} \texttt{Sym}(R), & \texttt{ASym}(R), & \texttt{Trans}(R), \\ \texttt{Ref}(R), & \texttt{Irr}(R), & \texttt{Dis}(R, S), \end{array}$$

where R,S are $\mathcal{SROIQ}$-roles such that $R, S \neq \mathbb{U}$. Given a role hierarchy $\mathcal{R}_h$, and a finite set of role assertions $\mathcal{R}_a$, the set of roles that are *simple* in $\mathcal{R} = \mathcal{R}_h \cup \mathcal{R}_a$ is inductively defined as follows:

- a role name is simple if it does not occur in the right-hand side of any inclusion in $\mathcal{R}_h$;
- R^{-} is simple if and only if so is R, for every $\mathcal{SROIQ}$-role R;
- if R occurs in the right-hand side of an inclusion in $\mathcal{R}_h$, R is simple if, for each inclusion $w \sqsubseteq R \in \mathcal{R}_h$, $w = S$ for some simple role S.

A set of role assertions $\mathcal{R}_a$ is simple if all roles R, S appearing in role assertions of the form $\texttt{Irr}(R)$, $\texttt{ASym}(R)$, or $\texttt{Dis}(R, S)$, are simple in $\mathcal{R}$.

A $\mathcal{SROIQ}$ *-RBox* $\mathcal{R}$ is a set $\mathcal{R} = \mathcal{R}_h \cup \mathcal{R}_a$, where $\mathcal{R}_h$ is a regular role hierarchy, and $\mathcal{R}_a$ is a simple set of role assertion.

Finally, a $\mathcal{SROIQ}$ *-knowledge base* consists of

- a $\mathcal{SROIQ}$-RBox $\mathcal{R} = \mathcal{R}_h \cup \mathcal{R}_a$;
- a finite set of inclusions of $\mathcal{SROIQ}$-concepts, in which the terms of the forms $\exists S.\texttt{Self}$, $(\leq n)R.C$, and $(\geq n)R.C$ are so that R is simple with respect to $\mathcal{R}$;
- a finite set of *individual assertions* of the forms $C(a)$, $R(a, b)$, $(\neg R)(a, b)$, and $a \neq b$, where a,b are individual names, C is a $\mathcal{SROIQ}$-concept, and R is a $\mathcal{SROIQ}$-role.

In the next section we mention another approach to augment the expressive power of description logics, which consists in extending description logic knowledge bases with SWRL rules.

3.5 SWRL Rules

In order to increase the expressive power of description logics, in [26] it was proposed to extend this framework with a simple form of Horn-style rules called SWRL rules. SWRL rules have the form

$$H \rightarrow B_1 \wedge \ldots \wedge B_n$$

where $H, B_1, \ldots, B_n$ are *atoms* of the forms $A(x), P(x, y), x = y, x \neq y$, with A a concept name, P a role name, and x, y either SWRL-variables or individual names.

A *binding* $\mathcal{B}(\mathcal{I})$ is any extension of the interpretation $\mathcal{I}$ which assigns a domain item to each SWRL-variable. An interpretation $\mathcal{I}$ satisfies a rule $H \rightarrow B_1 \wedge \ldots \wedge B_n$ if each binding $\mathcal{B}(\mathcal{I})$ which satisfies all the atoms $B_1, \ldots, B_n$ satisfies H also.

A description logic knowledge base $\mathcal{K}$ extended with a finite set of SWRL rules $\mathcal{R}$ is said to be consistent if and only if it admits a model which satisfies $\mathcal{K}$ and all the rules in $\mathcal{R}$.

The great expressive power of SWRL rules leads to undecidability if used in conjunction with the description logic $\mathcal{SROIQ}$. This was proved in [26] by providing a reduction to the consistency problem for $\mathcal{SROIQ}$-knowledge bases[5] of the *Tiling Problem*, a well-known undecidable problem studied in [4].

A similar undecidability result was proved in [28, Fact 4.2.2] with providing a reduction of the Post Correspondence problem. The same proof shows that undecidability is not caused by the great expressive power of $\mathcal{SROIQ}$, but just by the presence of existential quantifiers, in conjunction with SWRL rules.

[5]To be more precise, in [26] it was provided a reduction to the consistency problem in $\mathcal{SHOIN}$, a sublogic of $\mathcal{SROIQ}$.

Chapter 4

The $\forall^\pi$ Family

In this chapter we introduce the $\forall^\pi$ family of quantified fragments of set theory, consisting of three languages which allow the explicit manipulation of ordered pairs. We will provide a decision procedure for each language in this family. Furthermore, we will prove that, when the length of quantifier prefixes is bounded by a constant, the decision procedures for the three languages run in nondeterministic-polynomial time.

4.1 The Language $\forall^\pi_{\mathbf{0}}$

The language $\forall^\pi_{\mathbf{0}}$ is a quantified fragment of set theory which contains

- a denumerable infinity of *variables* *Vars*, defined in Section 2.1,
- the binary *pairing* operator $[\cdot, \cdot]$,
- the monadic function $\bar{\pi}(\cdot)$, which yields the collection of the non-pair members of its argument,
- the relators $\in$ and $=$,
- the Boolean connectives of propositional logic $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$,
- parentheses, and
- the universal quantifier $\forall$.

A *quantifier-free* $\forall^\pi_{\mathbf{0}}$*-formula* is any propositional combination of *atomic* $\forall^\pi_{\mathbf{0}}$*-formulae*. These are expressions of the following types:

$$x \in \bar{\pi}(z), \quad [x, y] \in z, \quad x = y,$$

with $x, y, z \in \mathit{Vars}$. Intuitively, terms of the form $[x, y]$ represent ordered pairs of sets. A *prenex* $\forall^\pi_{\mathbf{0}}$*-formula* is a formula of the form

$$Q_1 \cdots Q_n \psi$$

with $n \geq 0$, where ψ is a quantifier-free $\forall_{\mathbf{0}}^{\pi}$-formula, and the Q_i are restricted universal quantifier all of the form $(\forall x \in \bar{\pi}(y))$ or all of the form $(\forall [x, x'] \in y)$. We will refer to x and x' as *quantified variables* and to y as *domain variables*. A prenex $\forall_{\mathbf{0}}^{\pi}$-formula is said to be *simple* if no variable occurs both as a quantified and a domain variable, i.e., roughly speaking, no x, x' can be a y in the previous definition.

Finally, a $\forall_{\mathbf{0}}^{\pi}$-*formula* is any finite conjunction of simple prenex $\forall_{\mathbf{0}}^{\pi}$-formulae.

In order to provide the semantics definitions for $\forall_{\mathbf{0}}^{\pi}$, we extend set-theoretic interpretations to terms of the form $\bar{\pi}(x)$ as follows:

$$\mathbf{I}\bar{\pi}(x) \ =_{\text{Def}} \mathbf{I}x \setminus \pi_{\mathbf{I}}(\mathbf{I}x)$$

for every interpretation $\mathbf{I} = (M_{\mathbf{I}}, \pi_{\mathbf{I}})$, where $x \in \mathit{Vars}$.

According to Table 2.2, atomic $\forall_{\mathbf{0}}^{\pi}$-formulae are evaluated by a set-theoretic interpretation $\mathbf{I}$ as follows:

$$\begin{array}{lll} \mathbf{I} \models x \in \bar{\pi}(y) & \iff & \mathbf{I}x \in \mathbf{I}\bar{\pi}(y) \\ \mathbf{I} \models x = y & \iff & \mathbf{I}x = \mathbf{I}y \\ \mathbf{I} \models [x, y] \in z & \iff & \mathbf{I}[x, y] \in \mathbf{I}z. \end{array}$$

Evaluation of quantifier-free $\forall_{\mathbf{0}}^{\pi}$-formulae is carried out according to the standard rules of propositional logic, and simple prenex $\forall_{\mathbf{0}}^{\pi}$-formulae are evaluated as follows:

- $\mathbf{I} \models (\forall x \in \bar{\pi}(y))\varphi$ iff $\mathbf{I}' \models \varphi$ for every $\{x\}$-variant $\mathbf{I}'$ of $\mathbf{I}$ such that $\mathbf{I}'x \in \mathbf{I}\bar{\pi}(x)$,
- $\mathbf{I}(\forall [x, y] \in z)\varphi = \mathbf{true}$ iff $\mathbf{I}'\varphi = \mathbf{true}$ for every $\{x, y\}$-variant $\mathbf{I}'$ of $\mathbf{I}$ such that $\mathbf{I}'[x, y] \in \mathbf{I}z$.

Finally, an interpretation $\mathbf{I}$ evaluates to **true** a $\forall_{\mathbf{0}}^{\pi}$-formula $\varphi_1 \wedge \ldots \wedge \varphi_n$, where $n \in \mathbb{N}$ and φ_i are simple prenex $\forall_{\mathbf{0}}^{\pi}$-formulae, for $1 \leq i \leq n$, if and only if it evaluates to **true** all the conjuncts φ_i.

In the rest of this section we present and discuss a nondeterministic decision procedure for $\forall_{\mathbf{0}}^{\pi}$. We begin with introducing *skeletal representations*, which are finite structures which *represent* interpretations in a concise way, in a sense that will be clarified below. We will prove that skeletal representations can be used to witness the satisfiability of $\forall_{\mathbf{0}}^{\pi}$-formulae. In particular, the decision procedure presented here relies on the fact that any $\forall_{\mathbf{0}}^{\pi}$-formula is satisfiable if and only if there exists a size-bounded skeletal representation with some peculiar characteristics.

Later, the same problem will be considered from a complexity point of view, and we will show that it is NP-complete if we restrict to formulae whose quantifier prefixes have length bounded by a constant.

4.1.1 Skeletal representations

Skeletal representations are concise representations of set-theoretic interpretations, in the sense that, given an interpretation $\mathbf{I}$ and a finite collection of variables $x_1, \ldots, x_n$, one can construct a skeletal representation $\mathcal{S}$ which represents in a faithful and concise manner the relationships among the sets $\mathbf{I}x_i, \mathbf{I}[x_i, x_j]$, where $1 \leq i, j \leq n$.

Basic constituents of skeletal representations are atomic $\forall^{\pi}_{\mathbf{0}}$-formulae.

Given a finite collection $\mathcal{S}$ of atomic formulae, we denote with $\mathit{Vars}(\mathcal{S})$ the collection of the variables occurring in the formulae of $\mathcal{S}$. In addition, we indicate with $\in^+_{\mathcal{S}}$ (the *membership closure* of $\mathcal{S}$) the minimal transitive relation on $\mathit{Vars}(\mathcal{S})$ such that the following conditions hold:

- if "$x \in \bar{\pi}(y)$" $\in \mathcal{S}$, then $x \in^+_{\mathcal{S}} z$;
- if "$[x, y] \in z$" $\in \mathcal{S}$, then $x \in^+_{\mathcal{S}} z \wedge y \in^+_{\mathcal{S}} z$.

A finite collection $\mathcal{S}$ of atomic formulae is said to be a *skeletal representation* if $x \notin^+_{\mathcal{S}} x$, for all $x \in \mathit{Vars}(\mathcal{S})$.

Let $\mathcal{S}$ be a skeletal representation. We define the *height* of a variable $x \in \mathit{Vars}(\mathcal{S})$ with respect to $\mathcal{S}$ (which we write $\mathsf{height}_{\mathcal{S}}(x)$) as the length n of the longest $\in^+_{\mathcal{S}}$-chain of the form $x_1 \in^+_{\mathcal{S}} \ldots \in^+_{\mathcal{S}} x_n \in^+_{\mathcal{S}} x$ ending at x, with $x_1, \ldots, x_n \in \mathit{Vars}(\mathcal{S})$. Thus, $\mathsf{height}_{\mathcal{S}}(x) = 0$ if $y \notin^+_{\mathcal{S}} x$, for any $y \in \mathit{Vars}(\mathcal{S})$.

A skeletal representation $\mathcal{S}$ is said to be *V-extensional*, for a given set of variables V, if the following conditions hold:

- if "$x = y$" $\in \mathcal{S}$, then $x, y \in V$ and α^x_y and α^y_x belong to $\mathcal{S}$, for each atomic formula α in $\mathcal{S}$;
- if "$x = y$" $\notin \mathcal{S}$, for some $x, y \in V$, then the variables x and y must be explicitly *distinguished* in $\mathcal{S}$ either by some variable z, in the sense that "$z \in \bar{\pi}(x)$" $\in \mathcal{S}$ iff "$z \in \bar{\pi}(y)$" $\notin \mathcal{S}$, or by some pair $[z, z']$, in the sense that "$[z, z'] \in x$" $\in \mathcal{S}$ iff "$[z, z'] \in y$" $\notin \mathcal{S}$.

A skeletal representation can be turned into a special interpretation, namely its *realization*, in polynomial time as illustrated below. [1]

Definition 3 (Realization)**.** Let $\mathcal{S}$ be a skeletal representation, and let V and T be two finite and disjoint sets of variables such that $\mathit{Vars}(\mathcal{S}) \subseteq V \cup T$. In addition, let π be a pairing function, and let σ be an injective mapping which associates a set in the von Neumann hierarchy to each variable in T.

Then the *realization* $\mathcal{R} = (M_{\mathcal{R}}, \pi_{\mathcal{R}})$ of $\mathcal{S}$ relative to (V, T, π, σ) is defined by

$$\begin{array}{rcl} \pi_{\mathcal{R}} & =_{\text{Def}} & \pi \\ M_{\mathcal{R}}x & =_{\text{Def}} & \{\mathcal{R}y : \text{“}y \in \bar{\pi}(x)\text{”} \in \mathcal{S}\} \cup \{\mathcal{R}[y, z] : \text{“}[y, z] \in x\text{”} \in \mathcal{S}\} \cup s(x)\,, \end{array}$$

[1] Realizations were first introduced in [10], though with a slightly different meaning.

where

$$s(x) =_{\text{Def}} \begin{cases} \{\sigma(x)\} & \text{if } x \in T \\ \emptyset & \text{otherwise,} \end{cases}$$

for all $x \in \mathit{Vars}$.

It must be noticed that the recursion which defines the realization assignment is well posed, since being $\mathcal{S}$ a skeletal representation, it must be acyclic.

In the next lemma we show that, if π and σ are appropriately chosen, then the realization of a skeletal representation correctly models the atomic formulae in the representation itself. In more details, we will prove that realizations act as *minimal models* for skeletal representations, in the sense that if V, T are two disjoint sets of variables, $\mathcal{S}$ is a V-extensional skeletal representation such that $\mathit{Vars}(\mathcal{S}) \subseteq V \cup T$, and $\mathcal{R}$ is the realization of $\mathcal{S}$ relative to (V, T, π, σ), where π and σ were appropriately chosen, then $\mathcal{R} \models \alpha$ if and only if $\alpha \in \mathcal{S}$.

Lemma 4. *Let $\mathcal{S}$ be a skeletal representation, let V and T be two finite and disjoint collections of variables such that V is non empty, and $\mathit{Vars}(\mathcal{S}) \subseteq V \cup T$. Let π, σ be respectively a pairing function and an injective mapping from T to $\mathcal{V}$. Finally, let $\mathcal{R}$ be the realization of $\mathcal{S}$ with respect to (V, T, π, σ).*

Now let us suppose that the following conditions hold:

(RC1) *$\mathcal{S}$ is V-extensional;*

(RC2) *$\pi(u, v) \neq \mathcal{R}x$ for all $x \in V \cup T$, and for all $u, v \in \mathcal{V}$. (i.e. $\mathcal{R}$ does not assign any pair to the variables in $V \cup T$);*

(RC3) *$\sigma(t) \neq \pi(u, v)$ for all $t \in T$ and for all $u, v \in \mathcal{V}$ (i.e. σ does not contain pairs in its range);*

(RC4) *$\mathcal{R}x \neq \sigma(t)$ for all $x \in V \cup T, t \in T$.*

Then

(R1) *$\mathcal{R}x = \mathcal{R}y$ if and only if either "$x = y$" $\in \mathcal{S}$ or x and y coincide,*

(R2) *$\mathcal{R}x \in \mathcal{R}\bar{\pi}(y)$ if and only if "$x \in \bar{\pi}(y)$" $\in \mathcal{S}$, and*

(R3) *$\mathcal{R}[x, y] \in \mathcal{R}z$ if and only if "$[x, y] \in z$" $\in \mathcal{S}$*

for all $x, y, z \in V \cup T$.

Proof. To prove **(R1)** we reason as follows. Let x, y be two variables in $V \cup T$, and let us assume that "$x = y$" $\in \mathcal{S}$. Then x, y must be in V, since $\mathcal{S}$ is V-extensional. Now we prove that, in this case, $\mathcal{R}x$ must be a subset of $\mathcal{R}y$. Thus let u be a set arbitrarily chosen in $\mathcal{R}x$. From Definition 3 it follows that either $u = \mathcal{R}z$, for some $z \in V \cup T$ such that "$z \in \bar{\pi}(x)$" $\in \mathcal{S}$,

or $u = \mathcal{R}[z', z'']$, for some $z', z'' \in V \cup T$ such that "$[z', z''] \in x$" $\in \mathcal{S}$. In the former case, "$z \in \bar{\pi}(y)$" $\in \mathcal{S}$ follows from the V-extensionality of $\mathcal{S}$, so that $u = \mathcal{R}z \in \mathcal{R}y$. For the same reason, if "$[z', z''] \in x$" occurs in $\mathcal{S}$, then "$[z', z''] \in y$" must occurs in $\mathcal{S}$ also, so that $u = \mathcal{R}[z', \mathcal{R}z''] \in \mathcal{R}y$. Hence $u \in \mathcal{R}x$ entails $u \in \mathcal{R}y$, for all $u \in \mathcal{R}x$, so that $\mathcal{R}x$ must be a subset of $\mathcal{R}y$. Analogously it can be proved that $\mathcal{R}y \subseteq \mathcal{R}x$, and thus we can conclude that $\mathcal{R}x = \mathcal{R}y$ must hold in the case "$x = y$" $\in \mathcal{S}$.

For the converse direction we have to prove that if $\mathcal{R}x = \mathcal{R}y$, for distinct variables $x, y \in V \cup T$, then "$x = y$" $\in \mathcal{S}$. So, assume that "$x = y$" $\notin \mathcal{S}$, for two distinct variables $x, y \in V \cup T$ and consider first the case in which either x or y, say y, is a variable in T. From the definition of realization it follows that $\sigma(y) \in \mathcal{R}y$, while from **(RC3)** and **(RC4)** it follows that $\sigma(y) \notin \mathcal{R}x$, unless $x \in T$ and $\sigma(y) = \sigma(x)$. But in such a case, we would have $\sigma(x) = \sigma(y)$ and therefore x and y must coincide, since we supposed σ injective, contradicting our initial assumption that x and y are distinct variables. Therefore we have $\mathcal{R}x \neq \mathcal{R}y$.

Next, let us assume that $x, y \in V$. We will induct on

$$\max(\mathsf{height}_{\mathcal{S}}(x), \mathsf{height}_{\mathcal{S}}(y)).$$

From the V-extensionality of $\mathcal{S}$ it follows that x, y are distinguished in $\mathcal{S}$ by a variable z or by a pair $[z', z'']$. Let us first assume that x, y are distinguished in $\mathcal{S}$ by a variable z. If "$z \in \bar{\pi}(x)$" $\in \mathcal{S}$ and "$z \in \bar{\pi}(y)$" $\notin \mathcal{S}$, then for all w such that "$w \in \bar{\pi}(y)$" $\in \mathcal{S}$ we have $\mathcal{R}z \neq \mathcal{R}w$ by the inductive hypothesis, since $\mathsf{height}_{\mathcal{S}}(z) < \mathsf{height}_{\mathcal{S}}(x)$ and $\mathsf{height}_{\mathcal{S}}(w) < \mathsf{height}_{\mathcal{S}}(y)$. Furthermore, from **(RC2)** it follows also that $\mathcal{R}z \neq \mathcal{R}[w, \mathcal{R}w']$, for all w, w' such that "$[w, w'] \in y$" $\in \mathcal{S}$. Thus $\mathcal{R}z \in \mathcal{R}x \setminus \mathcal{R}y$. If "$z \in \bar{\pi}(y)$" $\in \mathcal{S}$ and "$z \in \bar{\pi}(x)$" $\notin \mathcal{S}$ we can prove that $\mathcal{R}z \in \mathcal{R}y \setminus \mathcal{R}x$ in an analogous way. In both cases we have $\mathcal{R}x \neq \mathcal{R}y$. On the other hand, if x, y are distinguished by a pair $[z', z'']$, we can argue as follows. Assume first that "$[z', z''] \in x$" $\in \mathcal{S}$ and "$[z', z''] \in y$" $\notin \mathcal{S}$. Plainly, $\mathcal{R}[z', z''] \in \mathcal{R}x$ so that, by **(RC2)**, $\mathcal{R}[z', z'']$ must coincide with $\mathcal{R}[w', w'']$, for some pair $[w', w'']$ such that "$[w', w''] \in y$" $\in \mathcal{S}$. Since π is a pairing function, we have $\mathcal{R}z' = \mathcal{R}w'$ and $\mathcal{R}z'' = \mathcal{R}w''$. Considering that $\max(\mathsf{height}_{\mathcal{S}}(z'), \mathsf{height}_{\mathcal{S}}(z'')) < \mathsf{height}_{\mathcal{S}}(x)$ and that $\max(\mathsf{height}_{\mathcal{S}}(w'), \mathsf{height}_{\mathcal{S}}(w'')) < \mathsf{height}_{\mathcal{S}}(y)$, the inductive hypothesis yields that

- z' and w' coincide or "$z' = w'$" is in $\mathcal{S}$, and
- z'' and w'' coincide or "$z'' = w''$" is in $\mathcal{S}$.

But then, by the V-extensionality of $\mathcal{S}$, "$[z', z''] \in y$" would be in $\mathcal{S}$, a contradiction. Hence, $\mathcal{R}[z', z''] \in \mathcal{R}x \setminus \mathcal{R}y$. Analogously, if "$[z', z''] \in x$" $\notin \mathcal{S}$ and "$[z', z''] \in y$" $\in \mathcal{S}$, we have $\mathcal{R}[z', z''] \in \mathcal{R}y \setminus \mathcal{R}x$. Therefore, in both cases we have $\mathcal{R}x \neq \mathcal{R}y$, proving **(R1)**.

Next we prove **(R2)**. If "$x \in \bar{\pi}(y)$" is an atomic formula in $\mathcal{S}$, then $\mathcal{R}x \in \mathcal{R}y$ directly follows from the definition of realization. Now let us

suppose "$x \in \bar{\pi}(y)$" $\notin \mathcal{S}$, and let us assume by way of contradiction that $\mathcal{R}x \in \mathcal{R}y$. We remark that $\mathcal{R}x$ cannot be a pair, in consequence of **(RC2)**, and, in the case $y \in T$, $\mathcal{R}x \neq \sigma(y)$ follows from **(RC4)**. Thus, from the definition of realization it follows that there must exists some variable z, distinct from x, such that "$z \in \bar{\pi}(y)$" $\in \mathcal{S}$, and $\mathcal{R}x = \mathcal{R}y$. Thus "$z = x$" $\in \mathcal{S}$ follows from **(R1)**, so that x, y must be in V, and "$x \in \bar{\pi}(y)$" must occur in $\mathcal{S}$, since $\mathcal{S}$ is V-extensional, thus contradicting our initial assumption "$x \in \bar{\pi}(y)$" $\notin \mathcal{S}$

Finally let us consider **(R3)**. Analogously to **(R2)**, "$[x, y] \in z$" $\in \mathcal{S}$ easily entails $\mathcal{R}[x, y] \in \mathcal{R}z$. Conversely, if $\mathcal{R}[x, y] \in \mathcal{R}z$, then, for **(RC2)** and **(RC3)**, there must exist two variables z', z'' such that "$[z', z'']$ $\in z$" occurs in $\mathcal{S}$, and $\mathcal{R}[z', z''] = \pi(\mathcal{R}z', \mathcal{R}z'') = \mathcal{R}[x, y] = \pi(\mathcal{R}x, \mathcal{R}y)$. However, $\mathcal{R}z' = \mathcal{R}x$ and $\mathcal{R}z'' = \mathcal{R}y$ must hold, as π is a pairing function. Thus, as consequence of **(R1)**, it must be either $z' = x$ or "$z' = x$" $\in \mathcal{S}$. Analogously, $z'' = y$ or "$z'' = x$" $\in \mathcal{S}$. In the case $z' = x, z' = y$, our thesis follows straightforwardly from "$[z', z'']$ $\in z$" $\in \mathcal{S}$. Concerning the other cases, it can easily be proved that "$[x, y] \in z$" follows if we consider the V-extensionality of $\mathcal{S}$. □

In fact, given $\mathcal{S}, V, T$ as in Definition 3, we can construct a pairing function π and a mapping σ so that they satisfy conditions **(RC2)**, **(RC3)**, and **(RC4)**. To this purpose, we begin with by introducing the following family $\{\pi_n\}_{n\in\mathbb{N}}$ of binary operations over sets, recursively defined by

$$\begin{aligned} \pi_0(u, v) &=_{\text{Def}} \{u, \{u, v\}\} \\ \pi_{n+1}(u, v) &=_{\text{Def}} \{\pi_n(u, v)\}, \end{aligned}$$

for every $u, v \in \mathcal{V}$. It can easily be proved that each function in this family is a pairing function.

Lemma 5. *Let π_n be defined as above. Then π_n is a pairing function.*

Proof. We proceed by induction on n. Let us consider the function π_0, and let u, v, u', v' be four sets in $\mathcal{V}$. Plainly, if $u = u'$ and $v = v'$, then $\pi_0(u, v) = \{u, \{u, v\}\} = \{u', \{u', v'\}\} = \pi_0(u', v')$. Conversely, let us assume that $\pi_0(u, v) = \pi_0(u', v')$. Then either one of the followings must hold:

(i) $u = u' \wedge v = v'$, or

(ii) $u = \{u', v'\} \wedge \{u, v\} = u'$.

But **(ii)** contradicts the regularity axiom of set theory, since in this case it would be $u = \{\{u, v\}, v'\}$. Thus **(i)** must hold. It remains to prove that, given any two sets u, v in the von Neumann hierarchy $\mathcal{V}$, their Cartesian product (with respect to π_0) is a set in $\mathcal{V}$. To this purpose, let us consider u, v any two sets in $\mathcal{V}$, and let us denote with $u \times_0 v$ the Cartesian product of u and v with respect to π_0. It is easy to see that $u \times_0 v$ is a subset of

$\mathcal{P}(u \cup v \cup \mathcal{P}(u \cup v))$. This is enough to say that $u \times_0 v \in \mathcal{V}$, as $u, v \in \mathcal{V}$ and the von Neumann hierarchy is closed under the powerset and the binary union operators.

Now let us consider the function π_{n+1}, with $n \in \mathbb{N}$, and let $u, v, u', v' \in \mathcal{V}$. Also in this case $\pi_{n+1}(u, v) = \pi_{n+1}(u', v')$ trivially follows from the definition of π_{n+1}, if $u = u'$ and $v = v'$. Thus let us assume that $\pi_{n+1}(u, v) = \pi_{n+1}(u', v')$, and prove that $u = u'$ and $v = v'$. However, $\pi_{n+1}(u, v) = \{\pi_n(u, v)\}$ and $\pi_{n+1}(u', v') = \{\pi_n(u', v')\}$, so that $\pi_n(u, v)$ and $\pi_n(u', v')$ must coincide. Then our thesis follows by applying the inductive hypothesis.

Concerning **(P2)**, it is enough to observe that, given $u, v \in \mathcal{V}$, $\pi_n(u', v') \in \mathcal{V}$ follows by applying the inductive hypothesis, for all $u' \in u$ and $v' \in v$, and then $\pi_{n+1}(u', v') = \{\pi_n(u', v')\}$ must be in $\mathcal{V}$ also. □

Next we show how to construct a suitable pairing function π and mapping σ, so as conditions **(RC2)**, **(RC3)**, and **(RC4)** of Lemma 4 are satisfied.

Lemma 6. *Let $\mathcal{S}$ be a skeletal representation, let V and $T = \{t_1, \ldots, t_m\}$ be two finite and disjoint sets of variables such that V is non empty and $Vars(\mathcal{S}) \subseteq V \cup T$. Then there exist a pairing function π and a mapping σ from T to $\mathcal{V}$, such that, if $\mathcal{S}$ is V-extensional, then **(R1)**, **(R2)**, and **(R3)** of Lemma 4 hold for the realization $\mathcal{R}$ of $\mathcal{S}$ relative to (V, T, π, σ).*

Proof. To prove the lemma, we provide a pairing function π and a mapping σ which satisfy **(RC2)**, **(RC3)**, and **(RC4)**. Thus, let us put

$$\begin{array}{rcl} \pi & =_{\text{Def}} & \pi_{|V|+|T|} \\ \sigma(t_i) & =_{\text{Def}} & \{k+1, k, i\}, \quad \text{for all } 0 \leq i < m, \end{array}$$

where $k = |V| \cdot (|V| + |T| + 3)$.[2] Then let $\mathcal{R}$ be the realization of $\mathcal{S}$ relative to (V, T, π, σ).

To prove **(RC2)**, we establish the more general property

$$\begin{array}{l} \text{if } \ \mathsf{height}_{\mathcal{S}}(x) \leq n \leq |V| + |T|, \ \text{ then } \ \mathcal{R}x \neq \pi_n(u, v), \\ \qquad \text{for all } x \in V \cup T \text{ and for all } u, v \in \mathcal{V}. \end{array} \tag{4.1}$$

Notice that, as V is non empty, then the sets assigned to the variables in T by σ must have cardinality exactly 3. On the other hand, $|\pi_0(u, v)| = 2$. and $|\pi_n(u, v)| = 1$ for all $u, v \in \mathcal{V}, n > 0$.

Let $n \leq |V| + |T|$ and let us assume by way of contradiction that $\mathcal{R}x = \pi_n(u, v)$ for some $u, v \in \mathcal{V}$ and some $x \in V \cup T$ of minimal height such that $0 \leq \mathsf{height}_{\mathcal{S}}(x) \leq n$. We can rule out at once the case in which $n = 0$, as in this case $\mathsf{height}_{\mathcal{S}}(x) = 0$, so that $|\mathcal{R}x| \leq 1$, and therefore $\mathcal{R}x \neq \pi_0(u, v)$, since $|\pi_0(u, v)| = |\{u, \{u, v\}\}| = 2$. Thus, we can assume that $n > 0$. Let

[2] We are assuming that integers are represented *à la* von Neumann, namely $0 =_{\text{Def}} \emptyset$ and, recursively, $n + 1 =_{\text{Def}} n \cup \{n\}$. In addition, we denote with $|\cdot|$ the *cardinality* operator over sets.

us consider first the case in which $\mathsf{height}_{\mathcal{S}}(x) = 0$. If $x \in V$ then, by the very definition of realization, we have $\mathcal{R}x = \emptyset \neq \pi_n(u, v)$. On the other hand, if $x \in T$, then $\mathcal{R}x = \{\{k+1, k, i\}\}$, for some $1 \leq i \leq |T|$, and since $|\{k+1, k, i\}| > |\pi_{n-1}(u, v)|$ and $\pi_n(u, v) = \{\pi_{n-1}(u, v)\}$, it follows that $\mathcal{R}x \neq \pi_n(u, v)$. In both cases we found a contradiction, so that we must have $\mathsf{height}_{\mathcal{S}}(x) > 0$.

On the other hand, if $\mathsf{height}_{\mathcal{S}}(x) > 0$, our absurd hypothesis $\mathcal{R}x = \pi_n(u, v) = \{\pi_{n-1}(u, v)\}$ and the definition of realization imply that either

(i) $\pi_{n-1}(u, v) = \{k+1, k, i\}$, for some $1 \leq i \leq |T|$, but provided that $x \in T$, or

(ii) $\pi_{n-1}(u, v) = \mathcal{R}y$, for some y such that "$y \in \bar{\pi}(x)$" $\in \mathcal{S}$, or

(iii) $\pi_{n-1}(u, v) = \mathcal{R}[y, z] = \pi(\mathcal{R}y, \mathcal{R}z)$, for some y, z such that "$[y, z] \in x$" $\in \mathcal{S}$.

We can exclude at once case **(i)**, since $|\pi_{n-1}(u, v)| \leq 2 < |\{k+1, k, i\}|$. Case **(ii)** can be excluded as well, since it would contradict the minimality of $\mathsf{height}_{\mathcal{S}}(x)$, as $\mathsf{height}_{\mathcal{S}}(y) < \mathsf{height}_{\mathcal{S}}(x)$. In case **(iii)**, from elementary properties of our pairing functions π_i it would follow that $|V| + |T| = n - 1$, contradicting our initial assumption that $n \leq |V| + |T|$. Thus (4.1) holds.

In view of (4.1), to establish **(RC2)** it is now enough to observe that $\mathsf{height}_{\mathcal{S}}(x) < |\mathit{Vars}(\mathcal{S})| \leq |V| + |T|$.

(RC3) easily follows if we observe that $|\sigma(t_i)| = 3$ and $|\pi_n(u, v)| \leq 2$, for all $t_i \in T, n \in \mathbb{N}, u, v \in \mathcal{V}$.

Next, since $\mathsf{rank}(\{k+1, k, i\}) = k + 2$, for $1 \leq i \leq |T|$ (as $k > |T|$), to establish **(RC4)** it will be enough to show that $\mathsf{rank}(\mathcal{R}x) \neq k + 2$, for $x \in V \cup T$. Thus, let $x \in V \cup T$. If $y \in^+_{\mathcal{S}} x$, for some $y \in T$, then $\mathsf{rank}(\mathcal{R}x) \geq \mathsf{rank}(\mathcal{R}y) \geq k + 3$. The same conclusion can be reached also in the case in which $x \in T$. On the other hand, if $y \notin^+_{\mathcal{S}} x$, for any $y \in T$ and $x \in V$, it can easily be proved that $\mathsf{rank}(\mathcal{R}x) \leq \mathsf{height}_{\mathcal{S}}(x) \cdot (|V| + |T| + 3) \leq |V| \cdot (|V| + |T| + 3) = k$, so that, in any case, $\mathsf{rank}(\mathcal{R}x) \neq k + 2$ holds, proving **(RC4)**. We proceed by induction on $\mathsf{height}_{\mathcal{S}}(x)$. Indeed, $\mathcal{R}x = \emptyset$ follows from the very definition of realization if $\mathsf{height}_{\mathcal{S}}(x) = 0$, so that $\mathsf{rank}(\mathcal{R}x) = 0 \leq |V| \cdot (|V| + |T| + 3)$ since we assumed V not empty. Now let us consider the case $\mathsf{height}_{\mathcal{S}}(x) > 0$. Then either one of the followings must hold:

(i) $\mathsf{rank}(\mathcal{R}x) = \mathsf{rank}(\mathcal{R}y) + 1$ for some y such that "$y \in \bar{\pi}(x)$" $\in \mathcal{S}$, or

(ii) $\mathsf{rank}(\mathcal{R}x) = \mathsf{rank}(\mathcal{R}[y, z]) + 1$ for some y, z such that "$[y, z] \in x$" $\in \mathcal{S}$.

If "$y \in \bar{\pi}(x)$" $\in \mathcal{S}$, then $\mathsf{height}_{\mathcal{S}}(y) < \mathsf{height}_{\mathcal{S}}(x)$, so that $\mathsf{rank}(\mathcal{R}y) \leq \mathsf{height}_{\mathcal{S}}(y) \cdot (|V| + |T| + 3)$ follows by applying the inductive hypothesis, and thus, if **(i)** holds, $\mathsf{rank}(\mathcal{R}x) = \mathsf{rank}(\mathcal{R}y) + 1 < \mathsf{height}_{\mathcal{S}}(y) \cdot (|V| + |T| +$

3) $<$ $\mathsf{height}_{\mathcal{S}}(x) \cdot (|V| + |T| + 3)$. Now let us suppose **(ii)**. $\mathsf{height}_{\mathcal{S}}(x) > \max(\mathsf{height}_{\mathcal{S}}(y), \mathsf{height}_{\mathcal{S}}(z))$ follows from the very definition of height, and thus $\mathcal{R}y \leq \mathsf{height}_{\mathcal{S}}(y) \cdot (|V| + |T| + 3)$ and $\mathcal{R}z \leq \mathsf{height}_{\mathcal{S}}(z) \cdot (|V| + |T| + 3)$ follows by applying the inductive hypothesis. In addition, $\mathsf{rank}(\mathcal{R}x) = \max(\mathsf{rank}(\mathcal{R}y), \mathsf{rank}(\mathcal{R}z)) + (|V| + |T| + 3) + 1$ holds, since we put $\pi = \pi_{|V|+|T|}$, and thus $\mathsf{rank}(\mathcal{R}x) \leq \max(\mathsf{height}_{\mathcal{S}}(y), \mathsf{height}_{\mathcal{S}}(z)) \cdot (|V| + |T| + 3) + (|V| + |T| + 3) + 1 = (\max(\mathsf{height}_{\mathcal{S}}(y), \mathsf{height}_{\mathcal{S}}(z)) + 1) \cdot (|V| + |T| + 3) + 1$, so that the thesis follows by observing that $\mathsf{height}_{\mathcal{S}}(x) > \max(\mathsf{height}_{\mathcal{S}}(y), \mathsf{height}_{\mathcal{S}}(z))$, as we assumed that "$[y, z] \in x$" $\in \mathcal{S}$. □

In light of Lemma 6, in the rest of this thesis we will omit to indicate π and σ when defining a realization, assuming that they are those indicated in Lemma 6.

Skeletal representations can be used as faithful witnesses for the satisfiability of $\forall_{\mathbf{0}}^{\pi}$-formulae, as shown in the decision procedure illustrated in the following section.

4.1.2 A Decision Procedure for $\forall_{\mathbf{0}}^{\pi}$

In this section we report a proof of the decidability of the s.p. for $\forall_{\mathbf{0}}$-formulae. This proof, and the consequent decidability result, represents the main contribution of the thesis, since from it most of the other results presented here are derived. This proof was originally presented in [12], with slight modifications. It follows much the same strategy of [8], reformulated in terms of skeletal representations, and suitably extended so as to cope with ordered pair terms.

First of all, it is convenient to recall the following notion, introduced in [10]. We say that a set Σ *distinguishes* a set S if $s \cap \Sigma \neq s' \cap \Sigma$ holds for any two distinct $s, s' \in S$. The following lemma will be used in the decidability proof provided later in this section.

Lemma 7.[3] *Any finite set S admits a set Σ which distinguishes it and such that $|\Sigma| \leq |S| - 1$.*

Proof. If $|S| \leq 1$, our claim is vacuously true. Otherwise, let $|S| > 1$ and, inductively, let us assume that our claim holds for any set S' such that $|S'| < |S|$. Then, pick $s \in S$. By our inductive hypothesis the set $S \setminus \{s\}$ admits a set Σ' which distinguishes it and such that $|\Sigma'| \leq |S| - 2$. If Σ' distinguishes S, we are done. Otherwise, there is an $s' \in S \setminus \{s\}$ such that $s \cap \Sigma' = s' \cap \Sigma'$. Let $d \in (s \setminus s') \cup (s' \setminus s)$ and consider $\Sigma =_{\text{Def}} \Sigma' \cup \{d\}$. We claim that Σ distinguishes S. Indeed, if this were not the case there would exist an $s'' \in S \setminus \{s, s'\}$ such that $s \cap \Sigma = s'' \cap \Sigma$, so that $s \cap \Sigma' = s'' \cap \Sigma'$ and, therefore, $s' \cap \Sigma' = s'' \cap \Sigma'$, contradicting our assumption that Σ' distinguishes $S \setminus \{s\}$. It only remains to observe that plainly $|\Sigma| \leq |S| - 1$. □

[3] See [10]. [38] also provides an extension to infinite sets.

In the following theorem we prove the decidability of the s.p. for $\forall^{\pi}_{\mathbf{0}}$-formulae by showing that skeletal representations can be used to witness satisfiability of $\forall^{\pi}_{\mathbf{0}}$-formulae.

Theorem 8. *Let φ be a $\forall^{\pi}_{\mathbf{0}}$-formula, and let $V = Vars(\varphi)$. Then φ is satisfiable iff there exists a V-extensional skeletal representation $\mathcal{S}$ such that:*

(i) $Vars(\mathcal{S}) \subseteq V \cup T$, for some T such that $|T| < 2|V|$;

(ii) $\mathcal{R} \models \varphi$, where $\mathcal{R}$ is the realization of $\mathcal{S}$ relative to (V, T).

Proof. To prove the theorem, it is enough to exhibit a skeletal representation $\mathcal{S}$ that satisfies conditions (i) and (ii) above, given a model $\mathbf{I}$ for φ.

Thus, let $\mathbf{I}$ be a model for φ and let $\Sigma = \{\mathbf{I}x : x \in V\}$. As shown in Lemma 7, there exists a collection Σ_0 of size strictly less than $|\Sigma|$ which witnesses all the inequalities among the members of Σ, in the sense that $s \cap \Sigma_0 \neq s' \cap \Sigma_0$ for any two distinct $s, s' \in \Sigma$. Let us *split* the pairs present in Σ_0 (relative to the pairing function $\pi_{\mathbf{I}}$ of $\mathbf{I}$) forming the collection

$$\Sigma_1 =_{\text{Def}} \Sigma_0 \setminus \pi_{\mathbf{I}}(\Sigma_0) \cup \bigcup \{\{u, v\} : \pi_{\mathbf{I}}(u, v) \in \Sigma_0\}.$$

Then we put $\Sigma_2 =_{\text{Def}} \Sigma_1 \setminus \Sigma$ and let T be any collection of variables in *Vars*, not already occurring in φ, such that $|T| = |\Sigma_2|$. Notice that $|T| \leq 2|\Sigma_0| < 2|V|$.

Finally, we define our skeletal representation as the collection $\mathcal{S}$ of atomic $\forall^{\pi}_{\mathbf{0}}$-formulae such that:

$$\begin{array}{lcl} \text{“}x \in \bar{\pi}(y)\text{”} \in \mathcal{S} & \iff & \mathbf{I}x \in \mathbf{I}\bar{\pi}(y) \\ \text{“}[x, y] \in z\text{”} \in \mathcal{S} & \iff & \mathbf{I}[x, y] \in \mathbf{I}z \\ \text{“}x = y\text{”} \in \mathcal{S} & \iff & \mathbf{I}x = \mathbf{I}y \ \text{ and } \ x, y \in V \end{array}$$

for all $x, y, z \in V \cup T$.

We begin with proving that the skeletal representation defined in this way is V-extensional, as required by condition (i). Plainly,

$$\text{“}x = y\text{”} \in \mathcal{S} \iff x, y \in V$$

follows from the very definition of $\mathcal{S}$. Now let α be an atomic formula in $\mathcal{S}$. $\alpha^y_x \in \mathcal{S}$, $\alpha^x_y \in \mathcal{S}$ trivially follow from the very definition of $\mathbf{I}$, as “$x = y$” $\in \mathcal{S}$ imply $\mathbf{I}x = \mathbf{I}y$. For example, let us consider the case $\alpha =$ “$x \in \bar{\pi}(z)$”, for some $z \in Vars(\mathcal{S})$. Then $\mathbf{I}x \in \mathbf{I}\bar{\pi}(z)$ follows from the very definition of $\mathbf{I}$, so that $\mathbf{I}y \in \mathbf{I}\bar{\pi}(z)$, and thus “$y \in \bar{\pi}(z)$” $\in \mathcal{S}$. To conclude that $\mathcal{S}$ is V-extensional, it remains to prove that, given two distinct variables $x, y \in V$, they are distinguished in $\mathcal{S}$ by some variable z, or by some pair $[z, z']$. To this purpose, we remark that Σ_0 distinguishes Σ, which contains both $\mathbf{I}x$ and $\mathbf{I}y$, so that there must exists $d \in \Sigma_1$ such that $d \in (\mathbf{I}x \setminus \mathbf{I}y) \cup (\mathbf{I}y \setminus \mathbf{I}x)$. Let us assume $d \in (\mathbf{I}x \setminus \mathbf{I}y)$ (the case $d \in (\mathbf{I}y \setminus \mathbf{I}x)$ is analogous). If d is

not a pair, with respect to the pairing function $\pi_{\mathbf{I}}$, then $d \in \Sigma_1$, so that $d \in \Sigma_2 \cup \Sigma$, which yields $\mathbf{I}z = d$, for some $z \in V \cup T$, and $\mathbf{I}z \in \mathbf{I}\bar{\pi}(x)$. Thus "$z \in \bar{\pi}(x)$" $\in \mathcal{S}$, and "$z \in \bar{\pi}(y)$" $\notin \mathcal{S}$ follow from the very definition of $\mathbf{I}$. Analogously, if $d = \pi_{\mathbf{I}}(d, d')$ for some sets d,d', then $d, d' \in \Sigma_2$, so that there must exist $z, z' \in V \cup T$ such that $\mathbf{I}z = d$ and $\mathbf{I}z' = d'$, and then it must be "$[z, z'] \in x$" $\in \mathcal{S}$ and "$[z, z'] \in y$" $\notin \mathcal{S}$.

We prove next that also condition (ii) is satisfied, i.e. $\mathcal{R} \models \varphi$ holds, where $\mathcal{R}$ is the realization of $\mathcal{S}$ relative to (V, T). This amounts to showing that $\mathcal{R}$ models correctly all conjuncts of φ. These are simple prenex $\forall^{\pi}_{\mathbf{0}}$-formulae whose free variables belong to $V \cup T$ and whose domain variables belong to V, which are correctly modeled by $\mathbf{I}$. It will therefore be enough to prove the following general property stating that

$$\mathbf{I} \models \psi \Longrightarrow \mathcal{R} \models \psi, \tag{4.2}$$

for every simple prenex $\forall^{\pi}_{\mathbf{0}}$-formula ψ such that $\mathit{Vars}(\psi) \subseteq V \cup T$ and whose domain variables, if any, belong to V.

We prove (4.2) by induction on the length of the quantifier prefix of ψ.

When ψ is quantifier-free, (4.2) follows from propositional logic, by observing that the definition of $\mathcal{S}$, together with Lemma 6, yields that $\mathbf{I}\alpha = \mathcal{R}\alpha$, for each atomic $\forall^{\pi}_{\mathbf{0}}$-formula α such that $\mathit{Vars}(\alpha) \subseteq V \cup T$.

For the inductive step, let ψ have either the form $(\forall x \in \bar{\pi}(y))\chi$ or the form $(\forall [x, y] \in z)\chi$, with χ a simple prenex $\forall^{\pi}_{\mathbf{0}}$-formula having one less quantifier than ψ. For the sake of simplicity, we consider here only the case in which ψ has the form $(\forall x \in \bar{\pi}(y))\chi$, as the other case can be dealt with much in the same manner. We remark that, by hypothesis, the domain variable y in $(\forall x \in \bar{\pi}(y))\chi$ belongs to V.

Let us assume that $\mathbf{I} \models \psi$. To complete the inductive proof of (4.2) we need to show that $\mathcal{R} \models \psi$. From $\mathbf{I} \models \psi$ it follows that $\mathbf{I} \models (w \in \bar{\pi}(y)) \rightarrow \chi^x_w$, for every variable w, and in particular for every variable $w \in W$, where $W =_{\text{Def}} \{w \in V \cup T : \text{"}w \in \bar{\pi}(y)\text{"} \in \mathcal{S}\}$. Let $w \in W$. We clearly have $\mathbf{I} \models w \in \bar{\pi}(y)$, and therefore $\mathbf{I} \models \chi^x_w$. Plainly, $\mathit{Vars}(\chi^x_w) \subseteq V \cup T$. In addition, all domain variables in χ^x_w belong to V, since this is the case for all domain variables in χ and w can not appear in χ^x_w as a domain variable, since x is a quantified variable of ψ and as such can not appear also as a domain variable in ψ, and therefore in χ. Hence, by inductive hypothesis, we have $\mathcal{R} \models \chi^x_w$ and, *a fortiori*, $\mathcal{R} \models (w \in \bar{\pi}(y)) \rightarrow \chi^x_w$.

Notice that the latter relation holds also for $w \in (V \cup T) \setminus W$, since in this case $\mathbf{I} \not\models (w \in \bar{\pi}(y))$ and therefore, as observed above, $\mathcal{R} \not\models (w \in \bar{\pi}(y))$. Thus we have

$$\mathcal{R} \models (w \in \bar{\pi}(y)) \rightarrow \chi^x_w, \tag{4.3}$$

for every $w \in V \cup T$. We show that (4.3) implies $\mathcal{R} \models (\forall x \in \bar{\pi}(y))\chi$, which is what we want to prove.

Indeed, if by contradiction $\mathcal{R} \not\models (\forall x \in \bar{\pi}(y))\chi$, then $\mathcal{R}' \not\models (x \in \bar{\pi}(y)) \rightarrow \chi$, for some $\{x\}$-variant $\mathcal{R}'$ of $\mathcal{R}$, so that $\mathcal{R}' \models (x \in \bar{\pi}(y))$ and $\mathcal{R}' \not\models \chi$. But

then

$$\mathcal{R}'x \in \mathcal{R}'\bar{\pi}(y) = \mathcal{R}\bar{\pi}(y) \subseteq \{\mathcal{R}z : \text{“}z \in \bar{\pi}(y)\text{”} \in \mathcal{S}\}.$$

Therefore $\mathcal{R}'x = \mathcal{R}z_0$, for some variable z_0 (in $V \cup T$) such that the literal “$z_0 \in \bar{\pi}(y)$” belongs to $\mathcal{S}$. Thus we have $\mathcal{R} \models z_0 \in \bar{\pi}(y)$ and $\mathcal{R} \not\models (z_0 \in \bar{\pi}(y)) \to \chi^x_{z_0}$, contradicting (4.3). Hence, $\mathcal{R} \models (\forall x \in \bar{\pi}(y))\chi$ holds, completing the inductive proof of (4.2) and, in turn, the proof of condition (ii) of the theorem. □

We observe that that condition (ii) of Theorem 8 is effectively verifiable. Indeed, if φ is a quantifier-free $\forall_{\mathbf{0}}$-formula, V is the set of the free variables occurring in φ, $\mathcal{S}$ is a V-extensional skeletal representation, and $\mathcal{R}$ is the realization of $\mathcal{S}$ relative to $(V, \mathit{Vars}(\mathcal{S}) \setminus V)$, then, in force of Lemma 3, it can be tested in polynomial time if $\mathcal{R}$ is a model for φ, since $\mathcal{R} \models \alpha$ if and only if α occurs in $\mathcal{S}$, for each atomic $\forall^{\pi}_{\mathbf{0}}$-formula α occurring in φ.

On the other hand, if φ is a simple prenex $\forall^{\pi}_{\mathbf{0}}$-formula, condition (ii) can be verified, as it will be proved in Lemma 9, by checking that $\mathcal{R}$ correctly models the *expansion* of φ relative to $\mathcal{S}$, which we define shortly. For a simple prenex $\forall^{\pi}_{\mathbf{0}}$-formula ψ, we put

$$\mathsf{exp}_{\mathcal{S}}(\psi) =_{\text{Def}} \begin{cases} \psi & \text{if } \psi \text{ is quantifier-free,} \\ \bigwedge\limits_{\text{“}x' \in \bar{\pi}(y)\text{”} \in \mathcal{S}} \mathsf{exp}_{\mathcal{S}}(\chi^x_{x'}) & \text{if } \psi = (\forall x \in \bar{\pi}(y))\chi, \\ \bigwedge\limits_{\text{“}[x',y'] \in z\text{”} \in \mathcal{S}} \mathsf{exp}_{\mathcal{S}}(\chi^{x,\,y}_{x',y'}) & \text{if } \psi = (\forall [x,y] \in z)\chi. \end{cases}$$

Then we put

$$\mathsf{Exp}_{\mathcal{S}}(\varphi) =_{\text{Def}} \mathsf{exp}_{\mathcal{S}}(\varphi_1) \wedge \ldots \wedge \mathsf{exp}_{\mathcal{S}}(\varphi_n),$$

where $\varphi_1, \ldots, \varphi_n$ are the (simple prenex) conjuncts of φ.

The next lemma shows the connection between realizations and expansions.

Lemma 9. *Let φ be a simple prenex $\forall_{\mathbf{0}}$-formula, let $V = \mathit{Vars}(\varphi)$, and let $\mathcal{S}$ be a V-extensional skeletal representation. Let $\mathcal{R}$ be the realization of $\mathcal{S}$ relative to $(V, \mathit{Vars}(\mathcal{S}) \setminus V)$. If $\mathcal{R} \models \mathsf{exp}_{\mathcal{S}}(\varphi)$ then $\mathcal{R} \models \varphi$.*

Proof. We proceed by induction on the length of the quantifier prefix of φ. If φ is quantifier-free, then the lemma trivially holds, since $\mathsf{exp}_{\mathcal{S}}(\varphi) = \varphi$ follows from the very definition of $\mathsf{exp}_{\mathcal{S}}(\cdot)$.

Now let $\varphi = (\forall x \in \bar{\pi}(y))\psi$, for some $x, y \in \mathit{Vars}$, and for some $\forall_{\mathbf{0}}$-formula ψ. In addition, let us assume that $\mathcal{R} \models \mathsf{exp}_{\mathcal{S}}(\varphi)$. It will be enough to prove that $\mathcal{R}' \models \psi$, for every $\mathcal{R}'$ $\{x\}$-variant of $\mathcal{R}$ such that $\mathcal{R}'x \in \mathcal{R}\bar{\pi}(y)$. In this case, $\mathcal{R}'x$ must coincide with $\mathcal{R}x'$, for some $x' \in \mathit{Vars}(\mathcal{S})$ such that “$x' \in \bar{\pi}(y)$” $\in \mathcal{S}$, since $\mathcal{R}y = \{\mathcal{R}z : \text{“}z \in \bar{\pi}(y)\text{”} \in \mathcal{S}\}$. In addition, x must be free in ψ, as quantifiers nesting is not allowed in $\forall_{\mathbf{0}}$-formulae. Then $\mathcal{R}' \models \psi$ if and only if $\mathcal{R} \models \psi^x_{x'}$, which, in turn, follows by applying the

inductive hypothesis, as $\mathsf{exp}_{\mathcal{S}}(\psi^x_{x'})$ is a conjunct of $\mathsf{exp}_{\mathcal{S}}(\varphi)$, and we assumed that $\mathcal{R} \models \mathsf{exp}_{\mathcal{S}}(\varphi)$. This allows us to conclude that $\mathcal{R} \models \varphi$, for every $\mathcal{R}'$ $\{x\}$-variant of $\mathcal{R}$ such that $\mathcal{R}'x \in \mathcal{R}\bar{\pi}(y)$, and then $\mathcal{R} \models \varphi$.

Analogously, if $\varphi = (\forall[x, y] \in z)\psi$, for some $x, y, z \in \mathit{Vars}$, and for some $\forall_{\mathbf{0}}$-formula ψ, then $R \models \mathsf{exp}_{\mathcal{S}}(\chi^{x,\,y}_{x',y'})$, for all variables x' and y' such that "$[x', y'] \in z$" occurs in $\mathcal{S}$, and then $\mathcal{R}' \models \chi$ follows by applying the inductive hypothesis, for each $\{x, y\}$-variant $\mathcal{R}'$ of $\mathcal{R}$ such that $\mathcal{R}'x = \mathcal{R}x'$ and $\mathcal{R}'y = \mathcal{R}y$, where "$[x', y'] \in z$". The thesis easily follows if we recall that the realization does not assign any pair to variables in $V \cup T$, so that $\pi_{\mathcal{R}}(\mathcal{R}z) = \{\mathcal{R}[x', y'] : \text{"}[x', y'] \in z\text{"} \in \mathcal{S}\}$. □

This allow us to conclude the decidability proof for the s.p. of $\forall_{\mathbf{0}}$-formulae.

Corollary 10. *The satisfiability problem for $\forall_{\mathbf{0}}$-formulae is decidable.*

Proof. As proved in Theorem 8, the satisfiability of a $\forall^{\pi}_{\mathbf{0}}$-formula φ can be tested by first guessing a V-extensional skeletal representation $\mathcal{S}$, where $V = \mathit{Vars}(\varphi)$, such that $|\mathit{Vars}(\mathcal{S})| < 3|\mathit{Vars}(\varphi)|$. It must be noticed that the number of possible V-extensional skeletal representations satisfying condition (i) of the theorem is finite. It can easily be verified that the size of such a skeletal representation $\mathcal{S}$ is polynomial in the size of φ.

In addition, condition (ii) of Theorem 8 is effectively verifiable, since, by Lemma 9, it corresponds to checking that the realization $\mathcal{R}$ of $\mathcal{S}$ relative to $(\mathit{Vars}(\varphi), \mathit{Vars}(\mathcal{S}) \setminus \mathit{Vars}(\varphi))$ correctly models all the conjuncts of $\mathsf{Exp}_{\varphi}(\mathcal{S})$, which is a quantifier-free $\forall^{\pi}_{\mathbf{0}}$-formula. □

In the next section we analyze the s.p. for $\forall_{\mathbf{0}}$-formulae from a complexity point of view.

4.1.3 Complexity Issues

The s.p. for propositional logic can be easily reduced to the one for $\forall^{\pi}_{\mathbf{0}}$-formulae as follows. Given a propositional formula Q, we construct in linear time a quantifier-free $\forall^{\pi}_{\mathbf{0}}$-formula ψ_Q, by replacing each propositional variable p in Q with a corresponding atomic formula $x_p \in \bar{\pi}(U)$, where U is a variable distinct from all variables x_p so introduced. It is then immediate to check that Q is propositionally satisfiable if and only if the resulting $\forall^{\pi}_{\mathbf{0}}$-formula ψ_Q is satisfiable. Thus the NP-hardness of the satisfiability of $\forall^{\pi}_{\mathbf{0}}$-formulae follows immediately.

On the other hand, if ℓ is the longest quantifier prefix of the conjuncts in φ, then it turns out that $|\mathsf{Exp}_{\mathcal{S}}(\varphi)| = \mathcal{O}(|\varphi|^{2\ell}) = \mathcal{O}(|\varphi|^{2\cdot|\varphi|})$, and therefore testing whether $\mathcal{R} \models \mathsf{Exp}_{\mathcal{S}}(\varphi)$ takes at most exponential time, as it amounts to check that the realization correctly models all the conjuncts of $\mathsf{Exp}_{\mathcal{S}}(\varphi)$, thus showing that the s.p. for $\forall^{\pi}_{\mathbf{0}}$-formula is in NExpTime.

Corollary 11. *The s.p. for $\forall^{\pi}_{\mathbf{0}}$-formula is in* NExpTime. □

However, the same proof shows that if we restrict to the collection of $\forall^{\pi}_{\mathbf{0}}$-formulae whose quantifier prefixes are bounded by a constant $h \geq 0$, which we call $(\forall^{\pi}_{0})^{\leq h}$, then $|\mathsf{Exp}_{\mathcal{S}}(\varphi)|$ is only polynomial in $|\varphi|$, for any $(\forall^{\pi}_{0})^{\leq h}$-formula φ, and therefore to test whether $\mathcal{R}$ models correctly $\mathsf{Exp}_{\mathcal{S}}(\varphi)$, and in turn to test whether $\mathcal{R} \models \varphi$, takes polynomial time in $|\varphi|$, proving the following result:

Corollary 12. *The s.p. for $(\forall^{\pi}_{0})^{\leq h}$-formulae is* NP*-complete, for any $h \geq 0$.* □

The language we just discussed was devised in [12] with the main goal of allowing the explicit manipulation of ordered pairs without breaking the NP-completeness of the decision problem. This was achieved by distinguishing between the pairs part from the non-pairs part of a set in the quantifier. In the next sections we examine two other different approaches for representing pairs in quantified fragments of set theory.

4.2 The Language $\forall^{\pi}_{\mathbf{0,2}}$

Now we introduce $\forall^{\pi}_{\mathbf{0,2}}$, in which, in contrast with $\forall^{\pi}_{\mathbf{0}}$, set variables and map variables belong to two different sorts. This language contains:

- the denumerable infinities of *set variables* $Vars_s$ and *map variables* $Vars_m$, both defined in Section 2.1;
- the binary *pairing* operator $[\cdot, \cdot]$;
- the relators $\in$ and $=$;
- the Boolean connectives of propositional logic $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$;
- parentheses, and
- the universal quantifier $\forall$.

Atomic $\forall^{\pi}_{\mathbf{0,2}}$ -formulae are expressions of the following types

$$x \in y, \quad x = y, \quad [x, y] \in f,$$

where x, y are set variables, and f is a map variable. *Quantifier-free* $\forall^{\pi}_{\mathbf{0,2}}$-formulae are Boolean combinations of atomic $\forall^{\pi}_{\mathbf{0,2}}$-formulae, while *prenex $\forall^{\pi}_{\mathbf{0,2}}$ -formulae* are those formulae of the following form:

$$Q_1 \dots Q_n \psi$$

with $n \in \mathbb{N}$, where ψ is a quantifier-free $\forall^{\pi}_{\mathbf{0,2}}$-formula, and $Q_1, \dots, Q_n$ are restricted quantifiers of the following forms

$$(\forall x \in y), \quad (\forall [x, x'] \in f)$$

with $x, y \in \mathit{Vars}_s$, $f \in \mathit{Vars}_m$. Analogously to $\forall^{\pi}_{\mathbf{0}}$, we will refer to x, x' as quantified variables, and to y, f as domain variables. A prenex $\forall^{\pi}_{\mathbf{0,2}}$-formula is said to be *simple* if nesting of quantifiers is not allowed, i.e. no variable can be occurred both as a domain and a quantified variable. Finally, $\forall^{\pi}_{\mathbf{0,2}}$-formulae are finite conjunctions of simple prenex $\forall^{\pi}_{\mathbf{0,2}}$-formulae.

Evaluation of atomic $\forall^{\pi}_{\mathbf{0,2}}$-formulae follows the usual rules for quantified fragments of set theory. In particular

$$\begin{array}{lll} \mathbf{I} \models x \in y & \iff & \mathbf{I}x \in \mathbf{I}y \\ \mathbf{I} \models x = y & \iff & \mathbf{I}x = \mathbf{I}y \\ \mathbf{I} \models [x, y] \in f & \iff & \mathbf{I}[x, y] \in \mathbf{I}f \end{array}$$

for all $x, y \in \mathit{Vars}_s$, and for each $f \in \mathit{Vars}_m$.

Let φ be a $\forall^{\pi}_{\mathbf{0,2}}$-formula. An interpretation $\mathbf{I}$ is said to be a *model* for φ if $\mathbf{I}$ is map-safe, and it evaluates φ to true.

It must be noticed that, since map variables and set variables have different sorts, we do not really need to be specific about the pairing function used by the interpretation. We clarify this in the next section, by introducing the notion of *map-isomorphism* among interpretations. Map-isomorphism will help us to enlight some relevant properties of the language $\forall^{\pi}_{\mathbf{0,2}}$.

4.2.1 Map-Isomorphic Interpretations

Given two map-safe interpretations $\mathbf{I}$ and $\mathbf{I}'$, we say that $\mathbf{I}$ and $\mathbf{I}'$ are *map-isomorphic*, and we write $\mathbf{I} \sim_m \mathbf{I}'$, if and only if

- $M_{\mathbf{I}}x = M_{\mathbf{I}'}x$, for every set variable x, and
- $\pi_{\mathbf{I}'}(u, v) \in M_{\mathbf{I}}f$ iff $\pi_{\mathbf{I}}(u, v) \in M_{\mathbf{I}'}f$, for each $u, v \in \mathcal{V}, f \in \mathit{Vars}_m$.

It can easily be verified that $\sim_m$ is an equivalence relation over map-safe interpretations. Indeed, each interpretation is map-isomorphic with itself, and thus $\sim_m$ is reflexive. The symmetricity of $\sim_m$ follows straightforwardly from the very definition of $\sim_m$. Finally, given any two $u, v \in \mathcal{V}$, a map variable f, and three map-safe interpretations $\mathbf{I}$, $\mathbf{I}'$,$\mathbf{I}''$ such that $\mathbf{I} \sim_m \mathbf{I}'$, $\mathbf{I}' \sim_m \mathbf{I}''$, then $\pi_{\mathbf{I}'}(u, v) \in M_{\mathbf{I}}f$ iff $\pi_{\mathbf{I}'}(u, v) \in M_{\mathbf{I}'}$, and $\pi_{\mathbf{I}'}(u, v) \in M_{\mathbf{I}'}$ iff $\pi_{\mathbf{I}''}(u, v) \in M_{\mathbf{I}''}$, so that $\pi_{\mathbf{I}'}(u, v) \in M_{\mathbf{I}}f$ if and only if $\pi_{\mathbf{I}'}(u, v) \in M_{\mathbf{I}'}$. Thus the transitivity of $\sim_m$ easily follows.

All the interpretations in the same equivalence class evaluate a $\forall^{\pi}_{\mathbf{0,2}}$-formula to the same truth value. In other words, each equivalence class uniquely identifies the truth value assigned to a formula by the interpretations it contains, as proved in the following lemma.

Lemma 13. *Let φ be a $\forall^{\pi}_{\mathbf{0,2}}$-formula, and let $\mathbf{I}$,$\mathbf{I}'$ two map-safe interpretations such that $\mathbf{I} \sim_m \mathbf{I}'$. Then*

$$\mathbf{I}\varphi \iff \mathbf{I}'\varphi.$$

Proof. Let $\mathbf{I} \models \varphi$, we have to prove that $\mathbf{I}' \models \varphi$. $\mathbf{I}$ evaluates to **true** each simple prenex $\forall^{\pi}_{\mathbf{0,2}}$-formula occurring as conjunct in φ, so that it will be enough to prove that, more in general,

$$\mathbf{I}\psi \Longrightarrow \mathbf{I}'\psi$$

for each simple prenex $\forall^{\pi}_{\mathbf{0,2}}$-formula ψ. We proceed by induction on the length of the quantifier prefix of ψ. To begin with, we observe that $\mathbf{I}$ and $\mathbf{I}'$ evaluate atomic $\forall^{\pi}_{\mathbf{0,2}}$-formulae in the same manner. Indeed, $\mathbf{I} \models x \in y \iff \mathbf{I}' \models x \in y$ and $\mathbf{I} \models x = y \iff \mathbf{I}' \models x = y$, for each $x, y \in \mathit{Vars}_s$, follow since $\mathbf{I}$ and $\mathbf{I}'$ are map-isomorphic, and hence they must coincide on set variables. Concerning formulae of the type $[x, y] \in f$, we recall that $\mathbf{I} \models [x, y] \in f$ if and only if $\pi_{\mathbf{I}}(M_{\mathbf{I}}x, M_{\mathbf{I}}y) \in M_{\mathbf{I}}f$. But $M_{\mathbf{I}}x = M_{\mathbf{I}'}x$ and $M_{\mathbf{I}}y = M_{\mathbf{I}'}y$, for any $x, y \in \mathit{Vars}_s$, as $\mathbf{I}$ and $\mathbf{I}'$ are map-isomorphic. Map-isomorphism guarantees also that $\pi_{\mathbf{I}}(M_{\mathbf{I}'}x, M_{\mathbf{I}'}y) \in M_{\mathbf{I}}f$ iff $\pi_{\mathbf{I}'}(M_{\mathbf{I}'}x, M_{\mathbf{I}'}y) \in M_{\mathbf{I}'}f$, and thus we can conclude that $\mathbf{I} \models [x, y] \in f$ if and only if $\mathbf{I}' \models [x, y] \in f$.

Thus, $\mathbf{I} \models \psi \iff \mathbf{I} \models \psi$ easily follows from propositional logic if ψ is quantifier-free.

Now let $\psi = (\forall x \in y)\chi$, for some $x, y \in \mathit{Vars}_s$, and for some simple prenex formula χ having one less quantifier than ψ. Given a set $u \in \mathcal{V}$, let us denote with $\mathbf{I}_u$ and ${\mathbf{I}'}_u$ the $\{x\}$-variants of $\mathbf{I}$ and $\mathbf{I}'$, respectively, such that $\mathbf{I}_u x = u, {\mathbf{I}'}_u x = u$. We observe that $\mathbf{I}_u$ and ${\mathbf{I}'}_u$ are still map-isomorphic, for every $u \in \mathcal{V}$, since they differs from $\mathbf{I}$ and $\mathbf{I}'$ just for the value assigned to a set variable, and the sets assigned to map variables remain unchanged. Thus, if $u \in \mathbf{I}'y = \mathbf{I}y$, ${\mathbf{I}'}_u \models \chi$ follows by applying the inductive hypothesis, since we assumed $\mathbf{I} \models (\forall x \in y)\chi$, which yields $\mathbf{I}_u \models \chi$. And this holds for every $u \in \mathbf{I}'y$, thus proving that $\mathbf{I}' \models (\forall x \in y)\chi = \varphi$.

Finally, let us consider the case $\psi = (\forall[x, y] \in f)\chi$, for some $x, y \in \mathit{Vars}_s$, $f \in \mathit{Vars}_m$, and for some simple prenex formula χ having one less quantifier than ψ. We proceed analogously to the previous case. Thus, given two sets $u, v \in \mathcal{V}$, we denote with $\mathbf{I}_{u,v}$ and ${\mathbf{I}'}_{u,v}$ the $\{x, y\}$-variants of $\mathbf{I}$ and $\mathbf{I}'$, respectively, such that $\mathbf{I}_{u,v}x = {\mathbf{I}'}_{u,v}x = u$ and $\mathbf{I}_{u,v}y = {\mathbf{I}'}_{u,v}y = v$. Plainly, $\mathbf{I}_{u,v}$ and ${\mathbf{I}'}_{u,v}$ are map-isomorphic, for every $u, v \in \mathcal{V}$, so that $\mathbf{I}_{u,v} \models \chi$ entails ${\mathbf{I}'}_{u,v} \models \chi$. Hence ${\mathbf{I}'}_{u,v} \models \chi$ follows, for every u, v such that $\pi_{\mathbf{I}'}(u, v) \in \mathbf{I}'f$, as $\mathbf{I} \models (\forall[x, y] \in f)\chi$, and then $\mathbf{I}' \models (\forall[x, y] \in f)\chi = \psi$. □

In addition, each equivalence class of $\sim_m$ contains at least one interpretation $\mathbf{I}^{\pi}$ for each possible pairing function π, such that $\pi_{\mathbf{I}^{\pi}} = \pi$. In fact, given any map-safe interpretation $\mathbf{I}$, and a pairing function π, an interpretation $\mathbf{I}^{\pi}$ such that $\mathbf{I}^{\pi} \sim_m \mathbf{I}$ and $\pi_{\mathbf{I}^{\pi}} = \pi$ can be easily constructed as follows:

$$\pi_{\mathbf{I}^{\pi}} =_{\text{Def}} \pi;$$

$$\begin{array}{lll} M_{\mathbf{I}^{\pi}}x =_{\text{Def}} M_{\mathbf{I}}x & & \text{for each } x \in \mathit{Vars}_s; \\ M_{\mathbf{I}^{\pi}}f =_{\text{Def}} \{\pi(u, v) : \pi_{\mathbf{I}}(u, v) \in M_{\mathbf{I}}x\} & & \text{for each } f \in \mathit{Vars}_m. \end{array}$$

Thus, given any pairing-function π, every satisfiable $\forall^{\pi}_{\mathbf{0,2}}$-formula has a model $\mathbf{I}$ such that $\pi_{\mathbf{I}} = \pi$.

The properties of models for $\forall^{\pi}_{\mathbf{0,2}}$-formulae we just stated allow us to prove the following lemma, which will be useful in proving the decidability of the s.p. for $\forall^{\pi}_{\mathbf{0,2}}$-formulae.

Lemma 14. *Every satisfiable $\forall^{\pi}_{\mathbf{0,2}}$-formula has a model which is pair-free with respect to set variables.*

Proof. Let φ be a satisfiable $\forall^{\pi}_{\mathbf{0,2}}$-formula, and let $\mathbf{I}$ be a model for φ. To prove the lemma, it will be enough to exhibit an interpretation $\mathbf{I}'$ such that $\mathbf{I}'$ is a model for φ, and $\pi_{\mathbf{I}'}(\mathbf{I}'x) = \emptyset$, for every set variable $x \in \mathit{Vars}_s$. To this purpose, we introduce the binary operation over sets π_Δ defined by

$$\pi_\Delta(u, v) =_{\text{Def}} \{\pi_{\mathbf{I}}(u, v), \Delta\}$$

for every $u, v \in \mathcal{V}$, where

$$\Delta =_{\text{Def}} \{\mathbf{I}x : x \in \mathit{Vars}_s\}.$$

It can easily be proved that π_Δ is a pairing function. Indeed, if u, v, u', v' are sets in the von Neumann hierarchy of sets $\mathcal{V}$, and $u = u', v = v'$, then $\pi_\Delta(u, v) = \pi_\Delta(u', v')$ follows from the very definition of π_Δ. Conversely, let us assume by contradiction that $\pi_\Delta(u, v) = \pi_\Delta(u', v')$ and one of $u \neq u'$, $v \neq v'$ holds. Then $\pi_{\mathbf{I}}(u, v) \neq \pi_{\mathbf{I}}(u', v')$, as $\pi_{\mathbf{I}}$ is a pairing function, so that $\pi_\Delta(u, v) = \{\pi_{\mathbf{I}}(u, v), \Delta\} = \{\pi_{\mathbf{I}}(u', v'), \Delta\} = \pi_\Delta(u', v')$ yields that $\pi_{\mathbf{I}}(u, v) = \Delta = \pi_{\mathbf{I}}(u', v')$, which is absurd. Thus property **(P1)** must hold for π_Δ. In addition, the Cartesian product of two sets u, v in $\mathcal{V}$ with respect to π_Δ must be another set in $\mathcal{V}$, since it is a subset of $\mathcal{P}(u \times_{\pi_{\mathbf{I}}} v \cup \Delta)$, where $\times_{\pi_{\mathbf{I}}}$ is the Cartesian product with respect to the pairing function $\pi_{\mathbf{I}}$. Thus, we can conclude that π_Δ is a pairing function.

Then φ must admit a model $\mathbf{I}'$ which is map-isomorphic with $\mathbf{I}$, and such that $\pi_{\mathbf{I}'} = \pi_\Delta$. In fact, given any model $\mathbf{I}$ of φ, our model $\mathbf{I}'$ such that $\mathbf{I} \sim_m \mathbf{I}'$ and $\pi_{\mathbf{I}'} = \pi_\Delta$ can be easily obtained from $\mathbf{I}$ by the construction process just illustrated above.

In addition, $\mathbf{I}'x = \mathbf{I}x$ does not contain any pair, for every set variable x, since this would contradict the regularity axiom. In fact, given any $x \in \mathit{Vars}_s$, $M_{\mathbf{I}'}x \in \Delta$ and $\Delta \in \pi_\Delta(u, v)$ follow from the very definition of π, for any two $u, v \in \mathcal{V}$. This allow us to conclude that $\pi_{\mathbf{I}'}(M_{\mathbf{I}'}x) = \emptyset$, thus proving lemma. □

The next section addresses the s.p. for $\forall^{\pi}_{\mathbf{0,2}}$-formulae.

4.2.2 A Decision Procedure for $\forall^{\pi}_{\mathbf{0,2}}$

In this section we solve the s.p. for $\forall^{\pi}_{\mathbf{0,2}}$-formulae by providing a reduction to the s.p. for $\forall^{\pi}_{\mathbf{0}}$-formulae. This reduction will allow us to deduce also some complexity information concerning this decision problem.

To begin with, it is convenient to introduce some further notations and definitions. Let φ be a $\forall^{\pi}_{\mathbf{0,2}}$-formula. We denote with $Vars_s(\varphi)$ the collection of set variables which occur free in φ, and with $Vars_m(\varphi)$ the collection of the map variables in φ (notice that map variables can not occurs as bounded variables in φ).

Next we define the following abbreviations for $\forall^{\pi}_{\mathbf{0}}$-formulae:

$$\begin{array}{rll} \mathsf{pair\_free}(x) & =_{\text{Def}} & (\forall [x', y] \in x)([x', y] \notin x) \\ \mathsf{is\_map}(x) & =_{\text{Def}} & (\forall x' \in \bar{\pi}(x))(x' \notin \bar{\pi}(x)). \end{array}$$

Finally, we introduce a mapping τ from $\forall^{\pi}_{\mathbf{0,2}}$-formulae to $\forall^{\pi}_{\mathbf{0}}$-formulae, defined by

$$\begin{array}{rll} \tau(\varphi) & =_{\text{Def}} & \bigwedge\limits_{x \in Vars_s(\varphi)} \mathsf{pair\_free}(x) \quad \wedge \\ & & \bigwedge\limits_{f \in Vars_m(\varphi)} \mathsf{is\_map}(f) \quad \wedge \\ & & \bigwedge\limits_{1 \leq i \leq n} \tau_0(\psi_i) \end{array}$$

where $\varphi = \psi_1 \wedge \ldots \wedge \psi_n$, for some $n \geq 1$ and for some simple prenex $\forall^{\pi}_{\mathbf{0,2}}$-formulae $\psi_1, \ldots, \psi_n$, and $\tau_0(\cdot)$ is a correspondence which associates to each simple prenex $\forall^{\pi}_{\mathbf{0,2}}$-formula ψ a corresponding simple prenex $\forall^{\pi}_{\mathbf{0}}$-formula $\tau_0(\psi)$, obtained from ψ replacing each quantifier of the type $(\forall x \in y)$ with $(\forall x \in \bar{\pi}(y))$, and each atomic formula of the type $x \in y$ with $x \in \bar{\pi}(y)$. It can easily be verified that, given any $\forall^{\pi}_{\mathbf{0,2}}$-formula φ, the corresponding $\forall^{\pi}_{\mathbf{0}}$-formula $\tau(\varphi)$ can be constructed in polynomial time, with respect to the size of φ. In order to show that the s.p. for $\forall^{\pi}_{\mathbf{0,2}}$-formulae is decidable, we first prove the following property of the mapping τ_0.

Lemma 15. *Let φ be a simple prenex $\forall^{\pi}_{\mathbf{0,2}}$-formula, and let $\mathbf{I}$ be an interpretation, which is pair-free with respect to the collection of the domain variables of φ. Then*

$$\mathbf{I} \models \varphi \iff \mathbf{I} \models \tau_0(\varphi).$$

Proof. We proceed by induction on the length of the quantifier prefix of φ. If φ is quantifier-free, then the lemma trivially holds, since $\mathbf{I}x \in \mathbf{I}\bar{\pi}(y)$ iff $\mathbf{I}x \in \mathbf{I}y$, as $\mathbf{I}$ is pair-free with respect to all the free variables $x, y \in Vars(\varphi)$.

Thus, let us suppose that $\varphi = (\forall x \in y)\chi$, for some $x, y \in Vars_s$, and for some simple prenex $\forall^{\pi}_{\mathbf{0,2}}$-formula χ with one less quantifier than φ. Again, $\mathbf{I}\bar{\pi}(y) = \mathbf{I}y$, since y is a domain variable of φ, and then $\pi_{\mathbf{I}}(\mathbf{I}y) = \emptyset$, as $\mathbf{I}$ is pair-free with respect to all the domain variables of φ. Thus, it will suffice to prove that

$$\mathbf{I}_u \models \chi \iff \mathbf{I}_u \models \chi \tag{4.4}$$

for all the sets $u \in \mathbf{I}y$, where $\mathbf{I}_u$ denotes the $\{x\}$-variant of $\mathbf{I}$ which assigns the set u to x. $\mathbf{I}_u$ is pair-free with respect to all the domain variables of χ, since $\mathbf{I}_u$ differs from $\mathbf{I}$ only for the value assigned to x, which is a quantified variable. Then (4.4) immediately follows by applying the inductive hypothesis to χ, for any $u \in \mathbf{I}y$, thus concluding the proof of this case.

Now, let us consider the case $\varphi = (\forall[x, y] \in f)\chi$, for some $x, y \in \mathit{Vars}_s$, $f \in \mathit{Vars}_m$, and for some simple prenex $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$-formula χ with one less quantifier than φ. It easily can be verified that

$$\mathbf{I}_{u,v} \models \chi \iff \mathbf{I}_{u,v} \models \chi,$$

for all u, v such that $\pi_{\mathbf{I}}(u, v) \in \mathbf{I}f$, where $\mathbf{I}_{u,v}$ indicates the $\{x, y\}$-variant of $\mathbf{I}$ such that $\mathbf{I}_{u,v}x = u$ and $\mathbf{I}_{u,v}y = v$. Again, x and y are quantified variables of φ, and thus $\mathbf{I}_{u,v}$ is still pair-free with respect the domain variables of φ. Hence, the thesis follows by applying the inductive hypothesis, since the domain variables of χ are a subset of those of φ. □

In view of the preceding lemma, it becomes very easy to show that any $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$-formula has exactly the same models of the corresponding $\forall^{\pi}_{\mathbf{0}}$-formula $\tau(\varphi)$. We provide a reduction of the s.p. for $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$-formulae to the s.p. of $\forall^{\pi}_{\mathbf{0}}$-formulae, thus proving the decidability of the decision problem of $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$.

Theorem 16. *The satisfiability problem for $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$-formulae is* NExpTime.

Proof. Let $\varphi = \psi_1 \wedge \ldots \wedge \psi_n$ be a $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$-formula. We prove that φ is satisfiable if and only if so is the $\forall^{\pi}_{\mathbf{0}}$-formula $\tau(\varphi)$. To begin with, let us assume that φ is satisfiable, and let $\mathbf{I}$ be a model of φ. We can assume without loss of generality that $\mathbf{I}$ is pair-free, as consequence of Lemma 14. Then $\mathbf{I}$ evaluates to **true** all the conjuncts of $\tau(\varphi)$ of the type $\mathsf{pair\_free}(x)$. In addition, $\mathbf{I}$ is map-safe, as it is a model for a $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$-formula, and hence it evaluates to **true** also all the conjuncts $\mathsf{is\_map}(f)$ of $\tau(\varphi)$. Finally, we observe that $\mathbf{I}$ is pair-free with respect to all the set variables, and thus it is pair-free also with respect to the domain variables of φ. Thus $\mathbf{I} \models \tau(\psi_i)$, for all $1 \leq i \leq n$, follows applying Lemma 15, and then $\mathbf{I}$ is a model for $\tau(\varphi)$, as it evaluates to true all its conjuncts.

Conversely, let $\mathbf{I}$ be a model for the $\forall^{\pi}_{\mathbf{0}}$-formula $\tau(\varphi)$. We can assume without loss of generality that $\mathbf{I}$ is map-safe, since it evaluates to **true** the conjuncts $\mathsf{is\_map}(f)$ of $\tau(\varphi)$, and then $\pi_{\mathbf{I}}(\mathbf{I}f) = \emptyset$ for all the map variables occurring in φ, while the other map variables are not relevant. In addition, it is pair-free with respect to all the set variables occurring free in φ, and thus also with respect to the subset of the domain variables of the conjuncts of φ. Then $\mathbf{I} \models \psi_i$, for $1 \leq i \leq n$, directly follows from Lemma 15, and this allow us to conclude definitively that $\mathbf{I}\tau(\varphi) \models \varphi$.

Finally, the thesis follows from Theorem 8, since the reduction we just provided is polynomial in the size of the tested formula φ. □

The reduction provided in the last theorem applies also if we restrict to $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$-formulae whose quantifier prefixes has length bounded by a constant h, which we refer to as $(\forall^{\pi}_{0})^{\leq h}$ *-formulae.*

Corollary 17. *The satisfiability problem for $(\forall^{\pi}_{0,2})^{\leq h}$-formulae is* NP*-complete, for any constant $h \in \mathbb{N}$.*

Proof. The s.p. for propositional formulae can be easily reduced to the s.p. for quantifier-free $\forall^{\pi}_{\mathbf{0,2}}$-formulae, which belong to the class of $(\forall^{\pi}_{0,2})^{\leq 0}$-formulae, by reasoning as in Theorem 4.1.3. Thus the s.p. for $(\forall^{\pi}_{0,2})^{\leq h}$-formulae, for all $h \geq 0$, is NP-hard.

In order to provide a complexity upper bound, it is enough to observe that the mapping τ associates to any $(\forall^{\pi}_{0,2})^{\leq h}$-formula φ a $(\forall^{\pi}_{0})^{\leq k}$-formula φ', with $k = \max(1, h)$. We remark that $\tau(\varphi)$ contains the pair_free and is_map conjuncts, which are formulae with one quantifier. For this reason, a quantifier-free $\forall^{\pi}_{\mathbf{0,2}}$-formula will be mapped by τ to a $(\forall^{\pi}_{0})^{\leq 1}$-formula. In any case, $k \leq h + 1$, so that the s.p. for $(\forall^{\pi}_{0,2})^{\leq h}$-formulae can be reduced in polynomial time to the s.p. for $(\forall^{\pi}_{0})^{\leq h+1}$-formulae. This allow us to conclude that the satisfiability of $(\forall^{\pi}_{0,2})^{\leq h}$-formulae can be tested in nondeterministic-polynomial time as consequence of Lemma 17. □

4.2.3 Some Remarks about $\forall_{\mathbf{0}}$

We conclude this section by observing that $\forall^{\pi}_{\mathbf{0,2}}$ strictly contains the language $\forall^{\pi}_{\mathbf{0}}$, presented in Section 2.3. Then all the decidability and complexity results reported in this section apply to $\forall_{\mathbf{0}}$ also, except for the NP-hardness of the satisfiability problem, which however can be easily proved by reasoning as in Theorem 4.1.3. Then the following corollaries directly follows from Theorem 16, and from Corollary 17. [4]

Corollary 18. *The satisfiability problem for $\forall_{\mathbf{0}}$-formulae is* NExpTime. □

Corollary 19. *The satisfiability problem for $(\forall_0)^{\leq h}$-formulae is* NP-*complete.* □

Up to now, we presented two quantified languages which allow the explicit manipulation of ordered pairs. They employ two different strategies to preserve the decidability: $\forall^{\pi}_{\mathbf{0}}$ introduces the *non-pairs* operator $\bar{\pi}(\cdot)$, which *extracts* the members of a set which are not pairs. However, it allows only universal quantification restricted either to pair members or to non-pair members of a set; on the other hand, $\forall^{\pi}_{\mathbf{0,2}}$ allows quantifiers of the form $(\forall x \in y)$, but it distinguishes between set and map variables, so that terms like $[f, f'] \in g$, with $f, f', g \in \mathit{Vars}_m$, are not allowed. This issue is overcome in the last language described in this chapter, called $\forall^{\pi}_{\mathbf{\Delta}}$, by introducing an additional condition on interpretations to be models.

4.3 The Language $\forall^{\pi}_{\mathbf{\Delta}}$

The language $\forall^{\pi}_{\mathbf{\Delta}}$ is a quantified fragment of set theory which contains:

[4] Note that these are not novel results, as it was already proved in [8]

- a denumerable infinity of variables $Vars = \{x, y, z, \ldots\}$;
- the binary pairing operator $[\cdot, \cdot]$;
- the relators $\in, =$;
- the Boolean connectives of propositional logic $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$;
- parentheses;
- the universal quantifier $\forall$.

Prenex $\forall^{\pi}_{\mathbf{\Delta}}$ *-formulae* are expressions of the form

$$(\forall x_1) \ldots (\forall x_n)\psi,$$

where $x_1, \ldots, x_n$ are variables and ψ is a propositional combination of *atomic* $\forall^{\pi}_{\mathbf{\Delta}}$*-formulae*, i.e., expressions of the following three types

$$x \in y, \quad x = y, \quad [x, y] \in z,$$

with $x, y, z \in Vars$. As before, formulae with an empty quantifier prefix (i.e., when $n = 0$) are said to be *quantifier-free*. Finally, a $\forall^{\pi}_{\mathbf{\Delta}}$*-formula* is a finite conjunction of prenex $\forall^{\pi}_{\mathbf{\Delta}}$-formulae.

For any $\forall^{\pi}_{\mathbf{\Delta}}$-formula φ, we will denote with $Vars(\varphi)$ the collection of variables occurring free (i.e., not bounded by any quantifier) in φ.

The semantics of $\forall^{\pi}_{\mathbf{\Delta}}$ differs from the semantics of the other set-theoretical languages presented in this thesis as it imposes some additional constraints to interpretations. In addition, interpretations are extended with an interpretation *domain* as follows.

A $\forall^{\pi}_{\mathbf{\Delta}}$*-interpretation* is a triple $I = (\Delta^I, M_I, \pi_I)$, where

- Δ^I, the interpretation *domain*, is a set of the von Neumann cumulative hierarchy of sets $\mathcal{V}$,
- M_I is an assignment such that $M_I x \in \Delta^I$ for each $x \in Vars$, and
- π_I is a pairing function.

A $\forall^{\pi}_{\mathbf{\Delta}}$-interpretation I *evaluates* any $\forall^{\pi}_{\mathbf{\Delta}}$-formula φ into a truth value $I\varphi \in \{\mathbf{true}, \mathbf{false}\}$ in the following recursive manner. First of all, evaluation of quantifier-free $\forall^{\pi}_{\mathbf{\Delta}}$-formulae is performed according to the standard rules of propositional logic, where the predicates $\in$ and $=$ are interpreted according to their standard meaning in set theory.

In order to extend recursively the evaluation also to quantified formulae, we extend the notion of W-variant to $\forall^{\pi}_{\mathbf{\Delta}}$-interpretations. Thus, let $I = (\Delta^I, M_I, \pi_I)$ and $J = (\Delta^J, M_J, \pi_J)$ be any two $\forall^{\pi}_{\mathbf{\Delta}}$-interpretations. We say that J is a W-variant of I if $\Delta^J = \Delta^I$, $\pi_J = \pi_I$, and M_J is a W-variant of M_I.

Then, recursively, we put $I(\forall x)\psi = \mathbf{true}$, provided that $J\psi = \mathbf{true}$, for every $\{x\}$-variant J of I.

Finally, we say that a $\forall^{\pi}_{\mathbf{\Delta}}$-interpretation I is a *model* for a $\forall^{\pi}_{\mathbf{\Delta}}$-formula φ (and write $I \models \varphi$) if the following conditions are satisfied:

(M1) the sets Δ^I and $\Delta^I \times \Delta^I$ are disjoint;

(M2) $M_I x \subseteq \Delta^I \cup (\Delta^I \times \Delta^I)$, for all $x \in \mathit{Vars}(\varphi)$;

(M3) I evaluates to **true** all the conjuncts of φ.

Thus, $\forall^{\pi}_{\mathbf{\Delta}}$-formula is said to be *satisfiable* if it admits a model, and then the satisfiability problem for $\forall^{\pi}_{\mathbf{\Delta}}$-formulae consists in determining whether a $\forall^{\pi}_{\mathbf{\Delta}}$-formula admits a model.

In the next section we solve the decision problem for our language $\forall^{\pi}_{\mathbf{\Delta}}$ by exploiting some correspondences between $\forall^{\pi}_{\mathbf{\Delta}}$ and the fragment $\forall^{\pi}_{\mathbf{0}}$.

4.3.1 A Decision Procedure for $\forall^{\pi}_{\mathbf{\Delta}}$

Firstly we show that the satisfiability problem for $\forall^{\pi}_{\mathbf{\Delta}}$-formulae is in NExpTime, by reducing it in polynomial time to the satisfiability problem for $\forall^{\pi}_{\mathbf{0}}$-formulae. Subsequently, we will examine the satisfiability problem for $(\forall^{\pi}_{\Delta})^{\leq h}$-formulae, namely $\forall^{\pi}_{\mathbf{\Delta}}$-formulae whose conjuncts have quantifier prefixes of length at most $h \geq 0$.

The satisfiability problem for $\forall^{\pi}_{\mathbf{\Delta}}$-formulae can be easily reduced to the satisfiability problem for $\forall^{\pi}_{\mathbf{0}}$-formulae. To this purpose, it is convenient to fix a variable $D \in \mathit{Vars}$, which we call the *knowledge domain variable*, and define a syntactic transformation $\tau_D(\cdot)$ on D-free $\forall^{\pi}_{\mathbf{\Delta}}$-formulae, namely $\forall^{\pi}_{\mathbf{\Delta}}$-formulae which contain no occurrence of the knowledge domain variable D. More specifically, given a D-free $\forall^{\pi}_{\mathbf{\Delta}}$-formula φ, then $\tau_D(\varphi)$ is obtained from φ by replacing each quantifier $(\forall x)$ in φ by its restricted form $(\forall x \in \bar{\pi}(D))$, and each atomic formula of the type $x \in y$ with $x \in \bar{\pi}(y)$. For the sake of simplicity, for the rest of the paper we will write $\tau(\cdot)$ in place of $\tau_D(\cdot)$ since the knowledge domain variable D will remain fixed. We also observe that, without any loss of generality, we can assume that any $\forall^{\pi}_{\mathbf{\Delta}}$-formula of interest is D-free.

The following lemma provides a useful semantical relation among $\forall^{\pi}_{\mathbf{\Delta}}$-formulae and their corresponding $\forall^{\pi}_{\mathbf{0}}$-formulae.

Lemma 20. *Let φ be a D-free prenex $\forall^{\pi}_{\mathbf{\Delta}}$-formula and let $\mathbf{I}$ and I be respectively a $\forall^{\pi}_{\mathbf{0}}$-interpretation and a $\forall^{\pi}_{\mathbf{\Delta}}$-interpretation satisfying the following conditions:*

(i) $\Delta^I = \mathbf{I}D$, $M_{\mathbf{I}} = M_I$, $\pi_{\mathbf{I}} = \pi_I$;

(ii) $\pi_{\mathbf{I}}(\mathbf{I}D) = \pi_I(\Delta^I) = \emptyset$.

Then $I\varphi = \mathbf{I}\tau(\varphi)$.

Proof. Let us denote with M and π the assignment $M_I = M_{\mathbf{I}}$ and the pairing function $\pi_I = \pi_{\mathbf{I}}$, respectively. We proceed by induction on the quantifier prefix length of the $\forall^{\pi}_{\mathbf{\Delta}}$-formula φ.

We observe that the set $Ix = \mathbf{I}x$ assigned to each variable x by both interpretations I and $\mathbf{I}$ we are considering can not be a pair, since $Ix \in \Delta^I$ and, by (ii), $\pi_I(\Delta^I) = \emptyset$. Thus $Mx \in My$ if and only if $Mx \in My \backslash \pi(My) = \mathbf{I}\bar{\pi}(y)$, so that $Ix \in Iy$ if and only if $\mathbf{I}x \in \mathbf{I}\bar{\pi}(y)$, for all $x, y \in \mathit{Vars}$. This allows us to conclude that I and $\mathbf{I}$ evaluate each atomic $\forall^{\pi}_{\mathbf{\Delta}}$-formula to the same truth value. Thus the thesis follows directly from propositional logic, when φ is quantifier-free.

Now let $\varphi = (\forall x)\chi$, for some $x \in \mathit{Vars} \setminus \{D\}$ and for some D-free $\forall^{\pi}_{\mathbf{\Delta}}$-formula χ. Let us denote with I_u and $\mathbf{I}_u$ the $\{x\}$-variant of I and of $\mathbf{I}$, respectively, such that $I_u x = u$ and $\mathbf{I}_u x = u$, for $u \in \mathcal{V}$. We must prove that $I_u\chi = \mathbf{true}$, for all $u \in \Delta^I$, if and only if $\mathbf{I}_u\tau(\chi) = \mathbf{true}$, for all $u \in \mathbf{I}\bar{\pi}(D)$.

Observe that, by (ii), $\mathbf{I}(\bar{\pi}(D)) = \mathbf{I}D = \Delta^I = MD$, and thus it will be enough to prove that

$$I_u\chi = \mathbf{true} \iff \mathbf{I}_u\tau(\chi) = \mathbf{true},$$

for all $u \in MD$. But this follows immediately by applying the inductive hypothesis to χ and to the interpretations I_u and $\mathbf{I}_u$, for $u \in MD$. □

It must be noticed that if I is a model of some $\forall^{\pi}_{\mathbf{\Delta}}$-knowledge base, then the condition $\pi_I(\Delta^I) = \emptyset$ (as required by the hypothesis (ii) of Lemma 20) is not a trivial consequence of **(M1)**, since this does not exclude that some pair $\pi_I(u, v)$ may belong to the domain Δ^I, for some $u, v \notin \Delta^I$. However, every consistent $\forall^{\pi}_{\mathbf{\Delta}}$-knowledge base admits a model with a *pair-free* domain, i.e. a model I such that $\pi_I(\Delta^I) = \emptyset$, as proved in the following lemma.

Lemma 21. *Let φ be a satisfiable $\forall^{\pi}_{\mathbf{\Delta}}$-formula. Then φ admits a model with a pair-free domain.*

Proof. Let I be a model for φ, and let J be the $\forall^{\pi}_{\mathbf{\Delta}}$-interpretation such that $\Delta^J = \Delta^I$, $M_J = M_I$, and

$$\pi_J(u, v) =_{\text{Def}} \begin{cases} \pi_I(u, v) & \text{if } u, v \in \Delta^I \\ \{\pi_I(u, v), \Delta^I\} & \text{otherwise,} \end{cases}$$

for all $u, v \in \mathcal{V}$.

We observe that J behaves in much the same way as I does, in the sense that $u \in Ix$ if and only if $u \in Jx$ and $\pi_I(u, v) \in Ix$ if and only if $\pi_J(u, v) \in Jx$, for all $u, v \in \Delta^J = \Delta^I$, since π_I and π_J coincide when restricted to the interpretation domain $\Delta^J = \Delta^I$. Thus $J \models \varphi$ follows directly from the assumption $I \models \varphi$.

It remains to prove that $\Delta^J = \Delta^I$ does not contain any pair. To this purpose, let $u, v \in \mathcal{V}$ be arbitrarily chosen. If $u, v \in \Delta^I$, then $\pi_J(u, v) \notin \Delta^I$ follows from **(M1)**, since $\pi_J(u, v) = \pi_I(u, v)$ in this case. Otherwise,

$\pi_J(u,v) = \{\pi_I(u,v), \Delta^I\}$ follows from the very definition of π_J, so that $\pi_J(u,v) \in \Delta^I$ would contradict the regularity axiom of set theory. In both cases we have $\pi_J(u,v) \notin \Delta^I$, and thus we can conclude that $\pi_J(\Delta^I) = \emptyset$. $\square$

In the following theorem we extend the transformation $\tau(\cdot)$ to (D-free) $\forall^\pi_{\mathbf{\Delta}}$-formulae, in order to complete the reduction of the s.p. for $\forall^\pi_{\mathbf{\Delta}}$-formulae to the satisfiability of $\forall^\pi_{\mathbf{0}}$-formulae.

Theorem 22. *The satisfiability problem for $\forall^\pi_{\mathbf{\Delta}}$-formulae is in* NExpTime.

Proof. We will prove that, given any $\forall^\pi_{\mathbf{\Delta}}$-formula φ, we can construct in polynomial time a corresponding $\forall^\pi_{\mathbf{0}}$-formula $\tau(\varphi)$ such that $\tau(\varphi)$ is satisfiable if and only if φ is consistent. Then the thesis will follow from Theorem 8. Let $\varphi = \psi_0 \wedge \ldots \wedge \psi_n$ be a $\forall^\pi_{\mathbf{\Delta}}$-formula, with $\psi_0, \ldots, \psi_n$ prenex $\forall^\pi_{\mathbf{\Delta}}$-formulae, and let $V = \mathit{Vars}(\varphi)$. Without loss of generality, we may assume that φ is D-free. We extend the transformation τ as follows:

$$\begin{aligned}\tau(\varphi) \quad =_{\text{Def}} \quad & \mathsf{pair\_free}(D) \wedge \bigwedge_{1 \le i \le n} \tau(\psi_i) \\ & \wedge \bigwedge_{z \in V} \big(z \in \bar{\pi}(D) \wedge z \subseteq D \cup (D \times D)\big),\end{aligned}$$

where

- $\mathsf{pair\_free}(D)$ stands for $(\forall [x,y] \in D)([x,y] \notin D)$, and
- $x \subseteq D \cup (D \times D)$ stands for

$$(\forall y \in \bar{\pi}(x))(y \in \bar{\pi}(D)) \wedge (\forall [y,z] \in x)([y,z] \in D)\,.$$

Let us first assume that $\tau(\varphi)$ admits a model $\mathbf{I}$. We construct a $\forall^\pi_{\mathbf{\Delta}}$-interpretation I by putting

$$\Delta^I =_{\text{Def}} \mathbf{I}D, \quad M_I =_{\text{Def}} M_{\mathbf{I}}, \quad \text{and} \quad \pi_I =_{\text{Def}} \pi_{\mathbf{I}}.$$

Observe that I satisfies conditions **(M1)**, **(M2)**, and **(M3)**. Indeed, from the first conjunct of $\tau(\varphi)$ we have $\pi_I(\Delta^I) = \emptyset$, so that $\Delta^I \cap (\Delta^I \times \Delta^I) = \emptyset$, i.e. **(M1)** is satisfied. Condition **(M2)** follows from the conjunction

$$z \in \bar{\pi}(D) \wedge z \subseteq D \cup (D \times D).$$

Finally, **(M3)** is a direct consequence of Lemma 20, since $\mathbf{I}(\mathsf{pair\_free}(D)) = \mathbf{true}$ entails $\pi_I(\Delta^I) = \pi_{\mathbf{I}}(\mathbf{I}D) = \emptyset$. Thus I is a model for φ.

For the converse direction, let I be a model for φ. In view of Lemma 21, we can assume without loss of generality that $\pi_I(\Delta^I) = \emptyset$. Let $\mathbf{I}$ be the $\forall^\pi_{\mathbf{0}}$-interpretation defined by

$$M_{\mathbf{I}}x =_{\text{Def}} \begin{cases} \Delta^I & \text{if } x = D \\ M_I x & \text{otherwise} \end{cases} \qquad \text{and} \qquad \pi_{\mathbf{I}} =_{\text{Def}} \pi_I,$$

for all $x \in \mathit{Vars}$. Plainly, $\mathbf{I}$ satisfies the conjunct $\mathsf{pair\_free}(D)$ of $\tau(\varphi)$, as we assumed $\pi_I(\Delta^I) = \emptyset$. In addition, it also satisfies the conjuncts $z \in \bar{\pi}(D)$ (by the properties of $\forall^{\pi}_{\mathbf{\Delta}}$-interpretations) and the conjuncts $z \subseteq D \cup (D \times D)$ (by **(M2)** and since $ID = \mathbf{I}\bar{\pi}(D)$), for $z \in V$. Finally, by Lemma 20, $\mathbf{I}$ satisfies $\tau(\psi_i)$, for each conjunct ψ_i of φ. Thus we can conclude that $\mathbf{I}$ is a model for $\tau(\varphi)$.

Hence, in order to check whether the knowledge base φ is consistent it is enough to check whether the corresponding $\forall^{\pi}_{\mathbf{0}}$-formula $\tau(\varphi)$ is satisfiable. As seen in Theorem 8, the latter check can be done in nondeterministic polynomial time, thus proving the theorem. □

Next we consider $(\forall^{\pi}_{\Delta})^{\leq h}$-formulae, namely $\forall^{\pi}_{\mathbf{\Delta}}$–formulae with conjuncts whose quantifier prefixes length is bounded by the constant $h \geq 0$.

By reasoning much as in the proof of Theorem 22, it is immediate to check that the satisfiability problem for any $(\forall^{\pi}_{\Delta})^{\leq h}$-formula φ can be reduced in polynomial time to the satisfiability problem of the corresponding $\forall^{\pi}_{\mathbf{0}}$-formula $\tau(\varphi)$ in $(\forall^{\pi}_{0})^{\leq h}$, and thus, by Corollary 4.1.3, it can be decided in nondeterministic polynomial time.

On the other hand, it is an easy matter to show that the satisfiability problem for $(\forall^{\pi}_{\Delta})^{\leq h}$-formulae is NP-hard. In fact, the proof we provided for the NP-hardness for $\forall^{\pi}_{\mathbf{0}}$ and $\forall^{\pi}_{\mathbf{0,2}}$ can be easily adapted to $\forall^{\pi}_{\mathbf{\Delta}}$.

The following corollary summarizes the above observations.

Corollary 23. *For any integer constant $h \geq 0$, the satisfiability problem for $(\forall^{\pi}_{\Delta})^{\leq h}$-formulae is* NP*-complete.* □

Chapter 5

Expressivity of the $\forall^\pi$ Languages

The languages in the $\forall^\pi$ family are expressive enough to include a large amount of set-theoretic constructs, in particular several map-related constructs like, for example, map restriction and map inverse. In this chapter we discuss the expressive power of this family by presenting the unquantified fragment of set theory $\mathsf{MLSS}^{\times}_{2,m}$,[1] and then providing a reduction to the s.p. problem for $(\forall^{\pi}_{0,2})^{\leq 2}$-formulae, thus proving that the s.p. for $\mathsf{MLSS}^{\times}_{2,m}$-formulae is NP-complete. One of the most relevant peculiarities of this language is the presence of the Cartesian product. Concerning this operator, we recall that in [9] it was proved that extending MLS with the Cartesian product and cardinality comparison leads to undecidability, while is actually unknown the decidability of MLS extended with the Cartesian product alone. On the other hand, $\mathsf{MLSS}^{\times}_{2,m}$ imposes some crucial restrictions on the usage of the map domain and range operators.

Then we study the applications of the languages in $\forall^\pi$ in the field of knowledge representation. Applications of Computable Set Theory to knowledge representation have been recently proposed in [14], where some correspondence between (decidable) fragments of set theory and description logics have been exploited. Here we recall a very expressive description logic, called $\mathcal{DL}\langle\forall^\pi\rangle$, which is the counterpart in the description logic framework of the set-theoretical language $\mathsf{MLSS}^{\times}_{2,m}$. A polynomial reduction of the consistency problem for $\mathcal{DL}\langle\forall^\pi\rangle$-knowledge bases to the satisfiability problem for $(\forall^{\pi}_{0,2})^{\leq 2}$-formulae is provided, thus proving that the consistency problem for $\mathcal{DL}\langle\forall^\pi\rangle$-knowledge bases is NP-complete.

Next we prove that this description logic can be extended with SWRL rules, thus identifying another fragment of the Semantic Web Ontology Language which can be extended with SWRL rules without disrupting the decidability.

Finally, we present a metamodeling-enabled version of $\mathcal{DL}\langle\forall^\pi\rangle$, and prove

[1] Here we present an extended version of the language studied in [13]

that the consistency problem in this novel description logic is still NP-complete.

5.1 The Language $\mathsf{MLSS}^{\times}_{2,m}$

$\mathsf{MLSS}^{\times}_{2,m}$ (Multi-Level Syllogistic with singleton, Cartesian product, and various map constructs, see [13]) is a two-sorted unquantified fragment of set theory which contains:

- a countably infinite collection of *set variables* $Vars_s = \{x, y, z, \ldots\}$;
- a countably infinite collection of *map variables* $Vars_m = \{f, g, h, \ldots\}$;
- the predicate symbols $\in$, $=$, $\subseteq$, $\mathsf{injective}(\cdot)$, $\mathsf{single\_valued}(\cdot)$, $\mathsf{bijective}(\cdot)$, $\mathsf{is\_reflexive}(\cdot)$, $\mathsf{is\_transitive}(\cdot)$, $\mathsf{is\_asym}(\cdot)$, $\mathsf{is\_irreflexive}\cdot$;
- the operator symbols $\cap$, $\cup$, $\setminus$, $\times$ (Cartesian product), $[\cdot,\cdot]$ (pair), $\{\cdot\}$ (singleton), $(\cdot)_{\cdot|}$, $(\cdot)_{|\cdot}$, $(\cdot)_{\cdot|\cdot}$ (map restriction operators), $(\cdot)^{-1}$ (map inverse), $\mathsf{sym}(\cdot)$ (symmetric closure), $\circ$ (composition of maps), $\mathsf{id}(\cdot)$ (identity),$\mathsf{dom}(\cdot)$ (domain), $\mathsf{range}(\cdot)$, $f[x]$ (map image);[2]
- parentheses (to construct compound terms);
- the logical connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ (to construct compound formulae).

Set and map terms allowed in this languages are those defined by the following syntax rules:

$$\begin{array}{rcl} X, Y & \longrightarrow & x \mid \emptyset \mid X \cup Y \mid X \cap Y \mid X \setminus Y \mid \{X\} \\ F, G & \longrightarrow & f \mid F \cup G \mid F \cap G \mid F \setminus G \mid \{[X, Y]\} \mid X \times Y \mid F^{-1} \mid \\ & & \mathsf{sym}(F) \mid F_{X|} \mid F_{|Y} \mid F_{X|Y} \mid \mathsf{id}(X) \end{array}$$

where x is a set variable, f is a map variable, X and Y are set terms, and F,G are map terms. $\mathsf{MLSS}^{\times}_{2,m}$ *-formulae* are Boolean combinations of *atomic* $\mathsf{MLSS}^{\times}_{2,m}$ *-formulae*, which extend the usual atomic formulae $X \in Y$, $X = Y$, $X \subseteq Y$, $[X, Y] \in F$, $F = G$, $F \subseteq G$ with predicates of the following forms

$$\begin{array}{llll} \mathsf{injective}(F), & \mathsf{single\_valued}(F), & \mathsf{bijective}(F), & \mathsf{is\_reflexive}(F), \\ \mathsf{is\_transitive}(F), & \mathsf{is\_asym}(F), & \mathsf{is\_irreflexive}(F), & F \circ F' \subseteq G, \\ \mathsf{dom}(F) \subseteq X, & \mathsf{range}(F) \subseteq X, & F[X] \subseteq Y & \end{array}$$

where F, F', G are map terms, and X, Y are set terms.

[2]Note that map domain, range and image operators are allowed in $\mathsf{MLSS}^{\times}_{2,m}$ only in a restricted way, as specified later.

We denote with $\mathsf{Terms}(\varphi)$ the collection of terms occurring in the $\mathsf{MLSS}^{\times}_{2,m}$-formula φ. For example,

$$\mathsf{Terms}(x \cup (y \cap y') \in z) = \{x \cup (y \cap y'), z\}.$$

The evaluation of atomic formulae of the types $\mathsf{injective}(\cdot)$, $\mathsf{single\_valued}(\cdot)$, $\mathsf{bijective}(\cdot)$, $\mathsf{is\_reflexive}(\cdot)$, $\mathsf{is\_transitive}(\cdot)$, $\mathsf{is\_asym}(\cdot)$, $\mathsf{is\_irreflexive}(\cdot)$ is reported in Table 2.2, while the evaluation of atomic $\mathsf{MLSS}^{\times}_{2,m}$-formulae of the remaining types is carried out as indicated below:

$$\begin{array}{rcl}
\mathbf{I} \models F \circ F' \subseteq G & \iff & (\forall u, v, v')(\pi_{\mathbf{I}}(u, v) \in \mathbf{I}F \wedge \pi_{\mathbf{I}}(v, v') \in \mathbf{I}F' \\
& & \qquad \rightarrow \pi_{\mathbf{I}}(u, v') \in \mathbf{I}G) \\
\mathbf{I} \models \mathsf{dom}(F) \subseteq X & \iff & (\forall u, v)(\pi_{\mathbf{I}}(u, v) \in \mathbf{I}F \rightarrow u \in \mathbf{I}X) \\
\mathbf{I} \models \mathsf{range}(F) \subseteq X & \iff & (\forall u, v)(\pi_{\mathbf{I}}(u, v) \in \mathbf{I}F \rightarrow v \in \mathbf{I}X) \\
\mathbf{I} \models F[X] \subseteq Y & \iff & (\forall u, v)(\pi_{\mathbf{I}}(u, v) \in \mathbf{I}F \wedge u \in \mathbf{I}X \rightarrow v \in \mathbf{I}Y).
\end{array}$$

Finally, the notions of model and satisfiability are defined as usual.

It must be noticed that $\mathsf{MLSS}^{\times}_{2,m}$ is a two-sorted extension of MLSS, as indicated by the subscript 2, while the decidability of the language obtained extending MLSS with the Cartesian product over generic variables is actually unknown.

In the next section we define a restricted set of $\mathsf{MLSS}^{\times}_{2,m}$-formulae, namely *normalized* $\mathsf{MLSS}^{\times}_{2,m}$*-formulae*, and we provide a reduction of the satisfiability problem for this kind of formulae to the s.p. for $(\forall^{\pi}_{0,2})^{\leq 2}$-formulae. Then we will prove that this result can be extended to general $\mathsf{MLSS}^{\times}_{2,m}$-formulae.

5.1.1 Normalized $\mathsf{MLSS}^{\times}_{2,m}$-Formulae

Normalized $\mathsf{MLSS}^{\times}_{2,m}$*-formulae* are finite conjunctions of positive and negative *normalized* $\mathsf{MLSS}^{\times}_{2,m}$*-literals* (in short literals), where positive normalized $\mathsf{MLSS}^{\times}_{2,m}$-literals (in short positive literals) are all those $\mathsf{MLSS}^{\times}_{2,m}$-formulae of the following types

$$\begin{array}{llll}
x \in y, & x = y \cup z, & x = y \setminus z, & f = g \cup h, \\
f = g \setminus h, & f = x \times y, & f = g_{x|}, & f = \mathsf{id}(x), \\
f \circ g \subseteq h, & \mathsf{is_reflexive}(f), & \mathsf{is_irreflexive}(f), &
\end{array}$$

with x, y, z set variables and f, g, h map variables, while negative normalized $\mathsf{MLSS}^{\times}_{2,m}$-literals (in short negative literals) are formulae of the form $\neg\gamma$, where γ is a positive literal.

In this section we present a polynomial reduction of the s.p. for normalized $\mathsf{MLSS}^{\times}_{2,m}$-formulae to the s.p. for $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$-formulae. In more details, given a normalized $\mathsf{MLSS}^{\times}_{2,m}$-formula, we show how to construct (in polynomial time) an equisatisfiable $(\forall^{\pi}_{0,2})^{\leq 2}$-formula (i.e. a $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$-formula with quantifier prefix lengths at most 2), thus proving that the s.p. for normalized $\mathsf{MLSS}^{\times}_{2,m}$-formulae is NP-complete.

To ease our reduction, we adopt an alternative syntax for $\forall^{\pi}_{\mathbf{0,2}}$-formulae by representing $\forall^{\pi}_{\mathbf{0,2}}$-formulae $\varphi = \psi_1 \wedge \ldots \wedge \psi_n$ as finite sets of simple-prenex $\forall^{\pi}_{\mathbf{0,2}}$-formulae $\{\psi_1, \ldots, \psi_n\}$.

In the next theorem we solve the satisfiability problem for normalized $\mathsf{MLSS}^{\times}_{2,m}$-formulae. Intuitively speaking, the reduction reported in the following proof is carried out by replacing, in the formula we are testing, each normalized $\mathsf{MLSS}^{\times}_{2,m}$-literal with a corresponding $\forall^{\pi}_{\mathbf{0,2}}$-formula, in such a way that the resulting formula is equisatisfiable.

Theorem 24. *The satisfiability problem for normalized* $\mathsf{MLSS}^{\times}_{2,m}$*-formulae can be decided in nondeterministic polynomial-time.*

Proof. Let $\varphi = \gamma_1 \wedge \ldots \wedge \gamma_n$ be a normalized $\mathsf{MLSS}^{\times}_{2,m}$-formula, with $\gamma_1, \ldots, \gamma_n$ normalized $\mathsf{MLSS}^{\times}_{2,m}$-literals. Let us define τ as the mapping which associate a $\forall^{\pi}_{\mathbf{0,2}}$-formula to each literal in φ as indicated in Table 5.1, where x, y, z are set variables, f, g, h are map variables, and w_i, w'_i, w''_i, for $1 \leq i \leq n$, are distinct set variables not occurring in φ.

Then the $(\forall^{\pi}_{0,2})^{\leq 2}$-formula φ', induced by φ, is defined as

$$\varphi' =_{\text{Def}} \bigcup_{1 \leq i \leq n} \tau\gamma_i.$$

It can easily be verified that, if $\mathbf{I}$ is a model for φ', then it is a model for φ also. Conversely, if $\mathbf{I}$ is a model for φ then

- it correctly models all the $\forall^{\pi}_{\mathbf{0,2}}$-formulae $\tau\gamma_i$, for every positive literals of γ_i of φ, and
- it can easily be extended (redefined) to the variables w_i, w'_i, w''_i so that it evaluates to **true** also the formulae $\tau\gamma_i$, when γ_i is a negative literal.

Then φ and φ' are equisatisfiable. Furthermore, $\varphi' \in (\forall^{\pi}_{0,2})^{\leq 2}$, so that the thesis of the theorem directly follows from Corollary 17. □

In the next section we extend this result to general $\mathsf{MLSS}^{\times}_{2,m}$-formulae.

5.1.2 A Decision Procedure for $\mathsf{MLSS}^{\times}_{2,m}$

In this section we prove that the satisfiability problem for (general) $\mathsf{MLSS}^{\times}_{2,m}$-formulae is NP-complete. The decision procedure presented in this section is a valid satisfiability test for MLS-formulae, since $\mathsf{MLSS}^{\times}_{2,m}$ includes MLS.

We begin with showing several steps which allow to *transform* a $\mathsf{MLSS}^{\times}_{2,m}$-formula into an equivalent one, in a form which will fit better our task. Then, we present a decision procedure for formulae of this form, which uses (as subprocedure) the one introduced in the proof of Theorem 24, and which runs in nondeterministic-polynomial time.

γ_i	$\tau\gamma_i$
$x \in y$	$\{x \in y\}$
$x = y \cup z'$	$\{(\forall x' \in x)(x' \in y \vee x' \in z'),$ $(\forall y' \in y)(y' \in x), (\forall z' \in z')(z' \in x)\}$
$\neg(x = y \cup z')$	$\{w_i \in x \wedge w_i \notin y \wedge w_i \notin z' \vee$ $w_i \notin x \wedge (w_i \in y \vee w_i \in z')\}$
$x = y \setminus z'$	$\{(\forall x' \in x)(x' \in y \wedge x' \notin z'),$ $(\forall y' \in y)(y' \in x \leftrightarrow y' \notin z')\}$
$\neg(x = y \setminus z)$	$\{w_i \in x \wedge (w_i \notin y \vee w_i \in z) \vee$ $w_i \in y \wedge w_i \notin z \wedge w_i \notin x\}$
$f = g \cup h$	$\{(\forall [x', y'] \in f)([x', y'] \in g \vee [x', y'] \in h),$ $(\forall [x', y'] \in g)([x', y'] \in f), (\forall [x', y'] \in h)([x', y'] \in f)\}$
$\neg(f = g \cup h)$	$\{[w_i, w_i'] \in f \wedge [w_i, w_i'] \notin g \wedge [w_i, w_i'] \notin h \vee$ $[w_i, w_i'] \notin f \wedge ([w_i, w_i'] \in g \vee [w_i, w_i'] \in h)\}$
$f = x \times y$	$\{(\forall x' \in x)(\forall y' \in y)([x', y'] \in f),$ $(\forall [x', y'] \in f)(x' \in x \wedge y' \in y)\}$
$\neg(f = x \times y)$	$\{[w_i, w_i'] \in f \wedge (w_i \notin x \vee w_i' \notin y) \vee$ $w_i \in x \wedge w_i' \in y \wedge [w_i, w_i'] \notin f\}$
$f = g_{x\|}$	$\{(\forall [x', y'] \in f)(x' \in x \wedge [x', y'] \in g),$ $(\forall [x', y'] \in g)(x' \in x \rightarrow [x', y'] \in f)\}$
$\neg(f = g_{x\|})$	$\{[wi, w_i'] \in f \wedge ([w_i, w_i'] \notin g \vee w_i \notin x) \vee$ $[w_i, w_i'] \in g \wedge w_i \in x \wedge [w_i, w_i'] \notin f\}$
$f = \mathsf{id}(x)$	$\{(\forall x' \in x)([x, x] \in f), (\forall [x', y'] \in x)(x' = y' \wedge x' \in x)\}$
$\neg(f = \mathsf{id}(x))$	$\{[w_i, w_i'] \in f \wedge (w_i \neq w_i' \vee w_i \notin x) \vee w_i \in x \wedge [w_i, wi] \notin f\}$
$f \circ g \subseteq h$	$\{(\forall [x', y'] \in f)(\forall [y'', z'] \in g)([x', z'] \in h)\}$
$\neg(f \circ g \subseteq h)$	$\{[w_i, w_i'] \in f \wedge [w_i', w_i''] \in g \wedge [w_i, w_i''] \notin h\}$
$\mathsf{is\_reflexive}(f)$	$\{(\forall [x', y'] \in f)([x', x'] \in f)\}$
$\neg(\mathsf{is\_reflexive}(f))$	$\{[w_i, w_i'] \in f \wedge w_i \neq w_i'\}$
$\mathsf{is\_irreflexive}(f)$	$\{(\forall [x', y'] \in f)(x' \neq y')\}$
$\neg(\mathsf{is\_irreflexive}(f))$	$\{[w_i, w_i] \in f\}$

Table 5.1: *The mapping τ*

Thus, let φ be a $\mathsf{MLSS}^{\times}_{2,m}$-formula. The terms of the forms $\emptyset$, $X \cap Y$, $F \cap G$, $\{[X,Y]\}$, $\mathsf{sym}(F)$, $F_{|X}$, and $F_{X|Y}$ can be eliminated from φ, obtaining an equivalent formula, in force of the following equivalences among sets:

$$\begin{aligned}
\emptyset &= u \setminus u \\
u \cap v &= (u \cup v) \setminus (u \setminus v) \setminus (v \setminus u) \\
\{[u,v]\} &= \{u\} \times \{v\} \\
\mathsf{sym}(m) &= m \cup m^{-1} \\
m_{u|v} &= m \cap (u \times v) \\
m_{|u} &= \left((m^{-1})_{u|} \right)^{-1}
\end{aligned}$$

for all $u, v, m \in \mathcal{V}$ such that m is a map. For the sake of conciseness, we are intentionally omitting any reference to the pairing function used, since all these equations are valid for any *fixed* pairing function.

Atomic $\mathsf{MLSS}^{\times}_{2,m}$-formulae of the types $X = Y$, $F = G$, $X \subseteq Y$, $F \subseteq G$, $[X,Y] \in F$, $\mathsf{single\_valued}(F)$, $\mathsf{bijective}(F)$, $\mathsf{is\_transitive}(F)$, $\mathsf{is\_asym}(F)$, $\mathsf{dom}(F) \subseteq X$, $\mathsf{range}(F) \subseteq X$, $F[X] \subseteq Y$ can be eliminated also, since the followings hold for all $u, v, m \in \mathcal{V}$ such that m is a map:

$$\begin{aligned}
u = v &\iff u = v \cup v \\
u \subseteq v &\iff v = v \cup u \\
[u,v] \in m &\iff \{u\} \times \{v\} \in m \\
\mathsf{single_valued}(m) &\iff \mathsf{injective}(m^{-1}) \\
\mathsf{bijective}(m) &\iff \mathsf{injective}(m) \wedge \mathsf{single_valued}(m) \\
\mathsf{is_transitive}(m) &\iff m \circ m \subseteq m \\
\mathsf{is_asym}(m) &\iff m \cap m^{-1} = \emptyset \\
\mathsf{dom}(m) \subseteq u &\iff m_{u|} = m \\
\mathsf{range}(m) \subseteq u &\iff \mathsf{dom}(m^{-1}) \subseteq u \\
m[u] \subseteq v &\iff \mathsf{range}(m_{u|}) \subseteq v.
\end{aligned}$$

Finally, *complex* terms can be *unfolded* by introducing a linear number of new variables. For example, if a term $x \cup (x \cup z)$ occurs in a formula φ, and φ' is the formula obtained from φ by replacing each occurrence of the term $x \cup z$ with the variable y, not already occurring in φ, then φ and $\varphi' \wedge y = x \cup z$ are equisatisfiable.

Thus, given any $\mathsf{MLSS}^{\times}_{2,m}$-formula φ, we can obtain an equivalent $\mathsf{MLSS}^{\times}_{2,m}$-formula φ' which is a Boolean combination of positive normalized $\mathsf{MLSS}^{\times}_{2,m}$-literals.[3] In addition, this transformation can be performed in linear time, with respect to the size of the formula. However, the resulting formula φ' is not necessarily a normalized $\mathsf{MLSS}^{\times}_{2,m}$-formula, since it may be not a simple conjunction of normalized literals, but a more complex propositional combination of formulae of this kind. If the resulting formula is not a $\mathsf{MLSS}^{\times}_{2,m}$-formula, its satisfiability can be tested in nondeterministic-polynomial time as follows.

[3]Of course, being a Boolean combination, it may contain also negative literals.

At first, we guess a *Boolean assignment* ν for the literals $\gamma_1, \ldots, \gamma_n$ which occur in φ', i.e an injective mapping from these formulae to $\{\mathbf{true}, \mathbf{false}\}$. Then we verify whether this assignment evaluates the formula φ' to **true**, which means checking that the formula obtained from φ' by replacing each atomic formula γ_i with $\nu\gamma_i$ is valid (indeed, this test can be performed in linear time). If not, we return a negative answer. Otherwise, we return **true** if and only if the normalized $\mathsf{MLSS}^{\times}_{2,m}$-formula φ'', defined below, is satisfiable. Let us denote with $neg(\cdot)$ the *negation* operator for normalized $\mathsf{MLSS}^{\times}_{2,m}$-literals, which, given any literal γ, returns $\neg\gamma$ if γ is positive, or γ' if $\gamma = \neg\gamma'$, for some positive literal γ', and let

$$\widehat{\gamma}_i = \begin{cases} \gamma_i & \text{if } \nu\gamma_i = \mathbf{true} \\ neg(\gamma_i) & \text{otherwise.} \end{cases}$$

Then φ'' is defined as

$$\varphi'' =_{\text{Def}} \bigwedge_{1 \leq n} \widehat{\gamma}_i.$$

The formula φ'' can be constructed from ν in linear time (with respect to the length of φ'), and the satisfiability test for φ'' requires nondeterministic-polynomial time (cf. Theorem 24). Thus, the decision procedure just presented runs in nondeterministic-polynomial time.

It can easily be proved that the test just outlined is sound and complete. Plainly, if $\mathbf{I}$ is a model for φ'', then

$$\mathbf{I} \models \gamma_i \iff \nu\gamma_i = \mathbf{true}, \tag{5.1}$$

for all $1 \leq i \leq n$, so that $\mathbf{I}$ must be a model for φ' also, since ν evaluates φ' to **true**. Conversely, any model $\mathbf{I}$ for φ' yields a Boolean assignment ν over the literals of φ'

$$\nu\gamma_i = \mathbf{true} \iff \mathbf{I} \models \gamma_i$$

which evaluates φ' to **true**, so that $\mathbf{I}$ correctly models the normalized formula φ'', obtained from φ' and ν as indicated above.

Finally, the NP-hardness of the s.p. for $\mathsf{MLSS}^{\times}_{2,m}$-formulae can be proved by reasoning as in Theorem 16, thus yielding the NP-completeness of the s.p. for $\mathsf{MLSS}^{\times}_{2,m}$.

Theorem 25. *The satisfiability problem for* $\mathsf{MLSS}^{\times}_{2,m}$*-formulae is* NP*-complete.* □

The language $\mathsf{MLSS}^{\times}_{2,m}$ allows one to express a very large amount of set-theoretic constructs, so that it provides a quite comprehensive vision of the expressive power of the quantified language $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$, since the s.p. for $\mathsf{MLSS}^{\times}_{2,m}$ can be reduced to the s.p. for $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$. In some sense, we can say that the language $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$ may be used as a definition language for higher-level set-theoretical languages.

It is noticeable that the great expressive power of $\forall^{\pi}_{\mathbf{0,2}}$ is not yet enough for some simple and very common map-related operators like domain, range and map image, so that these operators can not be freely used in $\mathsf{MLSS}^{\times}_{2,m}$, but only in a limited fashion. However, these restrictions are due to efficiency reasons, since dropping them would trigger the ExpTime-hardness of the satisfiability problem.

In the next section we prove this fact, by presenting a more general result which concerns MLS extensions which comprehend the map domain operator.

5.1.3 ExpTime-hardness of MLS with the image operator

In this section we provide a complexity lower bound for the satisfiability problem of the decidable extension MLS_{Im} of MLS with atomic formulae of the type $y = f[x]$. Our result is achieved by reducing the decision problem for the description logic $\mathcal{ALC}$, reviewed in Section 3.2, to the satisfiability problem for MLS_{Im}. In particular, we will show how the satisfiability problem for $\mathcal{ALC}$-concepts with respect to a finite set of inclusions can be reduced to the satisfiability of MLS_{Im}-formulae. Since $\exists R.C \equiv \neg(\forall R.(\neg C))$, without loss of generality we can consider only concepts that do not contain any occurrence of universal restriction.

Theorem 26. *The satisfiability problem for* MLS_{Im} *is* ExpTime*-hard.*

Proof. In view of Theorem 2, it is enough to exhibit a reduction from $\mathcal{ALC}$ to MLS_{Im}. Thus, given a finite collection $\mathcal{T}$ of $\mathcal{ALC}$-inclusions and an $\mathcal{ALC}$-concept C, we show how to construct an MLS_{Im}-formula which is satisfiable if and only if the concept C is satisfiable w.r.t. $\mathcal{T}$.

With a small abuse of notation, for the sake of conciseness, we sometimes will write $[u, v] \in \mathbf{I}F$ instead of $\pi_{\mathbf{I}}(u, v) \in \mathbf{I}F$ to indicate that the ordered pair $[u, v]$ is a member of $\mathbf{I}F$.

Let $\mathsf{Cpts} \subseteq \mathcal{N}^c$ and $\mathsf{Rls} \subseteq \mathcal{N}^r$ be the collections of the concept names and of the role names, respectively, occurring in C and in $\mathcal{T}$. Additionally, let τ be a function that injectively associates every concept name in Cpts to a set variable of the language MLS_{Im} and every role name in Rls to a map variable of MLS_{Im}. The function τ extends naturally to concepts and constraints in the following recursive way:

$$\begin{aligned}
\tau\top &=_{\text{Def}} U \\
\tau\bot &=_{\text{Def}} \emptyset \\
\tau(\neg C) &=_{\text{Def}} U \setminus \tau C \\
\tau(C \sqcap D) &=_{\text{Def}} \tau C \cap \tau D \\
\tau(C \sqcup D) &=_{\text{Def}} \tau C \cup \tau D \\
\tau(\exists R.C) &=_{\text{Def}} (\tau R)[\tau C] \\
\tau(C \sqsubseteq D) &=_{\text{Def}} \tau C \subseteq \tau D \\
\tau(C \equiv D) &=_{\text{Def}} \tau C = \tau D\,,
\end{aligned}$$

where U is a set variable of MLS_{Im} not in $\tau[\mathsf{Cpts}]$.

Let $\varphi =_{\text{Def}} \psi_1 \wedge \psi_2 \wedge \psi_3$ be the MLS_{Im}-formula in which:

$$\begin{array}{lll} \psi_1 & =_{\text{Def}} & U \neq \emptyset \wedge \bigwedge\limits_{A \in \mathsf{Cpts}} \tau A \subseteq U \wedge \bigwedge\limits_{R \in \mathsf{Rls}} (\tau R)[U] \subseteq U \\ \psi_2 & =_{\text{Def}} & \bigwedge\limits_{\gamma \in \mathcal{T}} \tau\gamma \\ \psi_3 & =_{\text{Def}} & \tau C \neq \emptyset. \end{array}$$

We observe that the size of the MLS_{Im}-formula φ is linear in the total size of $\mathcal{T}$ and C.

Next we show that φ is satisfiable (relative to the semantics of MLS_{Im}) if and only if C is satisfiable w.r.t. $\mathcal{T}$ (relative to the semantics of $\mathcal{ALC}$).

To begin with, let us assume that φ is satisfiable, and let $\mathbf{I}$ be a model for φ. We construct a descriptive interpretation $\mathcal{I}$, induced by $\mathbf{I}$, with domain $\Delta^{\mathcal{I}} =_{\text{Def}} \mathbf{I}U$, by putting

$$\begin{array}{lll} A^{\mathcal{I}} & =_{\text{Def}} & \mathbf{I}(\tau A) \\ R^{\mathcal{I}} & =_{\text{Def}} & \Big((\mathbf{I}(\tau R))_{\mathbf{I}U|}\Big)^{-1}, \end{array}$$

for every concept name A and role name R occurring in $\mathcal{T}$ or in C.[4]. Otherwise, the action of the interpretation $\mathcal{I}$ over the remaining concept and role names can be defined arbitrarily, as long as the constraints $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ hold, for each concept name A and role name R not occurring in $\mathcal{T}$ or in C.

Notice that since $\mathbf{I}$ models correctly all the literals in ψ_1, then we actually have $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for all concept names A and role names R, respectively, showing that $\mathcal{I}$ is a valid interpretation. Moreover, for every concept D involving only variables that occur in $\mathcal{T}$ and in C we have

$$D^{\mathcal{I}} = \mathbf{I}(\tau D). \tag{5.2}$$

We prove (5.2) by structural induction on the concept D. If D is of type $\top$, $\bot$, $\neg D'$, $D' \sqcup D''$, or $D' \sqcup D''$, then (5.2) follows directly from the definition of $\mathcal{I}$. Thus, the only interesting case occurs when the concept D is of type $(\exists R.D_0)$, with R a role name in Rls and D_0 a concept structurally simpler than D.

Let us show that $(\exists R.D_0)^{\mathcal{I}} = \mathbf{I}(\tau(\exists R.D_0))$. Let $v \in (\exists R.D_0)^{\mathcal{I}}$. Then there is a $u \in D_0^{\mathcal{I}} = \mathbf{I}(\tau D_0)$ such that

$$[v, u] \in R^{\mathcal{I}} = \Big((\mathbf{I}(\tau R))_{\mathbf{I}U|}\Big)^{-1}.$$

[4]Notice that we are assuming that ordered pairs are represented by means of the same pairing function by both the descriptive interpretation $\mathcal{I}$ and the set-theoretical one $\mathbf{I}$. However, this does not conflict with the semantics of $\mathcal{ALC}$, as in this language concept and role terms are strictly distinguished.

The latter implies $[u,v] \in (\mathbf{I}(\tau R))_{\mathbf{I}U|}$, so that $[u,v] \in \mathbf{I}(\tau R)$ (since $u \in \mathbf{I}U$), and therefore $v \in (\mathbf{I}(\tau R))[\mathbf{I}(\tau D_0)] = \mathbf{I}(\tau(\exists R.D_0))$. Hence, $(\exists R.D_0)^{\mathcal{I}} \subseteq \mathbf{I}(\tau(\exists R.D_0))$.

To show the converse inclusion, let now $v \in \mathbf{I}(\tau(\exists R.D_0))$. Then $v \in (\mathbf{I}(\tau R))[\mathbf{I}(\tau D_0)]$, so that $[u,v] \in \mathbf{I}(\tau R)$ for some $u \in \mathbf{I}(\tau D_0)$. Therefore $[u,v] \in (\mathbf{I}(\tau R))_{\mathbf{I}U|}$ (since by inductive hypothesis $\mathbf{I}(\tau D_0) = D_0^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} = \mathbf{I}U$ and therefore $u \in \mathbf{I}U$). Hence,

$$[v,u] \in \Big((\mathbf{I}(\tau R))_{M(U)|}\Big)^{-1} = R^{\mathcal{I}}.$$

And since $u \in D_0^{\mathcal{I}}$, then $v \in (\exists R.D_0)^{\mathcal{I}}$. Therefore $\mathbf{I}(\tau(\exists R.D_0)) \subseteq (\exists R.D_0)^{\mathcal{I}}$ which together with the previous inclusion yields $(\exists R.D_0)^{\mathcal{I}} = M(\tau(\exists R.D_0))$.

From (5.2) and the fact that $\mathbf{I}$ models correctly all the conjuncts of ψ_2, it follows that $\mathcal{I}$ is a model for $\mathcal{T}$. Additionally, since $\mathbf{I}$ satisfies ψ_3, it also follows that $\mathcal{I}$ satisfies C, so that the interpretation $\mathcal{I}$ induced by the model $\mathbf{I}$ satisfies C w.r.t. $\mathcal{T}$. This completes the first half of the proof.

Conversely, let $\mathcal{I}$ be a model for C w.r.t. $\mathcal{T}$. Without loss of generality, we may assume that $\Delta^{\mathcal{I}}$ is a set belonging to the von Neumann hierarchy $\mathcal{V}$ (otherwise, we embed $\Delta^{\mathcal{I}}$ in $\mathcal{V}$).

Given any fixed pairing function π, we construct a set-theoretic interpretation $\mathbf{I}$ induced by $\mathcal{I}$ as follows:

$$\begin{array}{rcl} \pi_{\mathbf{I}} & =_{\text{Def}} & \pi, \\ M_{\mathbf{I}}U & =_{\text{Def}} & \Delta^{\mathcal{I}}, \\ M_{\mathbf{I}}(\tau A) & =_{\text{Def}} & A^{\mathcal{I}}, \\ M_{\mathbf{I}}(\tau R) & =_{\text{Def}} & \{\pi(u,v) : [v,u] \in R^{\mathcal{I}}\}, \end{array}$$

for all concept names A and role names R occurring in $\mathcal{T}$ and in C (as usual, we do not need to be specific on the remaining variables of MLS_{Im}), and show that $\mathbf{I}$ is a model for φ.

Much as was done before, we prove by structural induction that

$$\mathbf{I}(\tau D) = D^{\mathcal{I}}\,, \tag{5.3}$$

for every concept D involving only concept and role names occurring in $\mathcal{T}$ and C. As before, the only relevant case to be considered is when D is of type $(\exists R.D_0)$. To prove (5.3) for a concept D of type $(\exists R.D_0)$, it is enough to show that $\mathbf{I}(\tau R)[\mathbf{I}(\tau D_0)] = (\exists R.D_0)^{\mathcal{I}}$.

Let $u \in (\exists R.D_0)^{\mathcal{I}}$. Then there is a $v \in D_0^{\mathcal{I}}$ such that $[u,v] \in R^{\mathcal{I}}$. Therefore $[v,u] \in \mathbf{I}(\tau R)$ and since by inductive hypothesis $v \in \mathbf{I}(\tau D_0)$, it follows that $u \in \mathbf{I}((\tau R)[\tau D_0])$. Hence we have $(\exists R.D_0)^{\mathcal{I}} \subseteq \mathbf{I}(\tau R)[\mathbf{I}(\tau D_0)]$.

To prove the converse inclusion, let $u \in \mathbf{I}(\tau R)[\tau D_0])$. Hence, there exists $v \in \mathbf{I}(\tau D_0)$ such that $[v,u] \in \mathbf{I}(\tau R)$. But then $[u,v] \in R^{\mathcal{I}}$ and since by inductive hypothesis $\mathbf{I}(\tau D_0) = D_0^{\mathcal{I}}$, we have $v \in D_0^{\mathcal{I}}$, and thus $u \in (\exists R.D_0)^{\mathcal{I}}$. Therefore we have $\mathbf{I}(\tau R)[\mathbf{I}(\tau D_0)] \subseteq (\exists R.D_0)^{\mathcal{I}}$, which together with the previously established inclusion yields $\mathbf{I}(\tau R)[\mathbf{I}(\tau D_0)] = (\exists R.D_0)^{\mathcal{I}}$.

Having established (5.3), it is immediate to check that the assignment $\mathbf{I}$ satisfies the MLS_{Im}-formula φ, completing the proof of the theorem. □

This complexity lower bound extends to all the MLS extensions in which the range and domain restriction operators can be used without the limitations imposed in $\mathsf{MLSS}^{\times}_{2,m}$. This can be easily proved in force of the set-theoretic equation $m[u] = \mathsf{range}(m_{u|})$, which holds for all sets u, m in the von Neumann hierarchy of sets $\mathcal{V}$ such that m is a map.

Analogously, extending MLS with literals of the types $x = \mathsf{dom}(f)$, $f = g_{x|}$, and $f = g^{-1}$ would trigger the ExpTime-hardness of the decision problem, since $\mathsf{range}(m) = \mathsf{dom}\,(m^{-1})$.

In this section we devised a correspondence between a description logic and a fragment of set theory to derive a complexity lower bound for fragments of set theory from a similar result presented in the description logics area. Conversely, in the next sections we will provide complexity upper bounds for some description logics by deriving them from the complexity results we obtained for the $\forall^{\pi}$ languages, presented earlier in this thesis.

This will allow us to claim that $\forall^{\pi}$ languages are expressive enough to be profitably employed as knowledge representation languages.

5.2 The Description Logic $\mathcal{DL}\langle\forall^{\pi}\rangle$

Description logics are a family of well-studied logic-based formalisms for knowledge representation, whose basic notions was reviewed earlier in Chapter 3. The descriptive semantics of these languages is set-theoretical (see Tables 3.1, 3.2, and 3.3), and this yields a straightforward mapping of most of the constructs used in these languages to the framework of Computable Set Theory (cf. [14]). In this chapter we show that $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$, and, consequently, also $\forall^{\pi}_{\mathbf{0}}$, is expressive enough to include the very expressive description logic $\mathcal{DL}\langle\forall^{\pi}\rangle$, originally presented in [12], which will be reviewed below.[5]

Let $\mathcal{N}^c, \mathcal{N}^r, \mathcal{N}^i$ be the three denumerable, infinite and mutually disjoint collections of concept, role, and individual names, respectively. $\mathcal{DL}\langle\forall^{\pi}\rangle$-concept terms and $\mathcal{DL}\langle\forall^{\pi}\rangle$-role terms are formed according to the following syntax rules:

$$\begin{array}{rcl} C, D & \longrightarrow & A \mid \top \mid \bot \mid \neg C \mid C \sqcup D \mid C \sqcap D \mid \{a\} \mid \exists R.\mathtt{Self} \mid \exists R.\{a\} \\ R, S & \longrightarrow & P \mid \mathbb{U} \mid R^{-} \mid \neg R \mid R \sqcup S \mid R \sqcap S \mid R_{C|} \mid R_{|D} \mid R_{C|D} \mid \\ & & \mathtt{id}(C) \mid \mathtt{sym}(R), \end{array}$$

where C, D denote $\mathcal{DL}\langle\forall^{\pi}\rangle$-concept terms, R, S denote $\mathcal{DL}\langle\forall^{\pi}\rangle$-role terms, A, P denote a concept and a role name, respectively, and a denotes an individual name. A $\mathcal{DL}\langle\forall^{\pi}\rangle$-knowledge base is then a finite collection of state-

[5] In [12] this description logic was presented with the name $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{0}}\rangle$.

ments of the following types:

$$\begin{array}{lllll} C \equiv D, & C \sqsubseteq D, & R \equiv S, & R \sqsubseteq S, & C \sqsubseteq \forall R.D, \\ \exists R.C \sqsubseteq D, & R \circ R' \sqsubseteq S, & \texttt{Trans}(R), & \texttt{Ref}(R), & \texttt{Irr}(R), \\ \texttt{ASym}(R), & C(a), & R(a,b), & & \end{array}$$

where C, D are $\mathcal{DL}\langle\forall^\pi\rangle$-concept terms, R, S, R' are $\mathcal{DL}\langle\forall^\pi\rangle$-role terms, and a, b are individual names.

Notice that the above definition of $\mathcal{DL}\langle\forall^\pi\rangle$ is not minimal, as we intended to give a clear and immediate overview of its expressive power.

$\mathcal{DL}\langle\forall^\pi\rangle$ does not allow number restrictions. Additionally, value restriction and existential quantification are restricted to the left-hand side and right-hand side of inclusions, respectively. Nevertheless, the set of allowed constructs is extremely large. In particular, complex role constructors can be used freely, in contrast with most expressive description logics (see, for example, the description logic $\mathcal{SROIQ}$, reviewed in Section 3.4). Despite of this, reasoning in $\mathcal{DL}\langle\forall^\pi\rangle$ is NP-complete, as will be proved in the following theorem.

Theorem 27. *The consistency problem for $\mathcal{DL}\langle\forall^\pi\rangle$-knowledge bases is* NP*-complete.*

Proof. We will show that the consistency problem for $\mathcal{DL}\langle\forall^\pi\rangle$-knowledge bases reduces to the satisfiability problem for $(\forall^\pi_{0,2})^{\leq 2}$-formulae.

We begin by observing that we can restrict our attention to $\mathcal{DL}\langle\forall^\pi\rangle$-knowledge bases containing only statements of the following types:

$$\begin{array}{lllll} A \equiv \top, & A \equiv \neg B, & A \equiv B \sqcup B', & A \equiv \{a\}, & A \equiv \exists P.\{a\}, \\ P \equiv \mathbb{U}, & P \equiv \neg Q, & P \equiv Q \sqcup Q', & P \equiv Q_{A|} & P \equiv Q^-, \\ P \equiv \texttt{id}(A), & P \circ P' \sqsubseteq Q, & \texttt{Ref}(P), & & \end{array}$$

where A, B, B' are concept names, P, P', Q, Q' are role names, and a is an individual name.

In fact, by reasoning much as in Section 5.1.2, any $\mathcal{DL}\langle\forall^\pi\rangle$-knowledge base $\mathcal{K}$ can be easily transformed into a knowledge base $\mathcal{K}'$, which contains only statements of these types, and such that $\mathcal{K}$ is consistent if and only if $\mathcal{K}'$ is, if we consider that

$$(\exists R.C)^{\mathcal{I}} = \mathsf{dom}\left(\left(R^{\mathcal{I}}\right)_{|C^{\mathcal{I}}}\right)$$

holds for every role R, for every concept C, and for every interpretation $\mathcal{I}$.

For the sake of completeness, we report the equations which justify this

transformation step.

$$
\begin{array}{rcl}
(\bot)^{\mathcal{I}} & = & (\neg\top)^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} & = & (\neg(\neg C \sqcup \neg D))^{\mathcal{I}} \\
(R \sqcap S)^{\mathcal{I}} & = & (\neg(\neg R \sqcup \neg S))^{\mathcal{I}} \\
(R_{|C})^{\mathcal{I}} & = & \left(\left((R^-)_{C|}\right)^-\right)^{\mathcal{I}} \\
(R_{C|D})^{\mathcal{I}} & = & \left(R_{C|} \sqcap R_{|D}\right)^{\mathcal{I}} \\
(\mathtt{sym}(R))^{\mathcal{I}} & = & (R \sqcup R^-)^{\mathcal{I}}
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{I} \models C \equiv D & \iff & \mathcal{I} \models C \equiv D \sqcup D \\
\mathcal{I} \models C \sqsubseteq D & \iff & \mathcal{I} \models D \equiv D \sqcup C \\
\mathcal{I} \models R \equiv S & \iff & \mathcal{I} \models R \equiv S \sqcup S \\
\mathcal{I} \models R \sqsubseteq S & \iff & \mathcal{I} \models S \equiv S \sqcup R \\
\mathcal{I} \models C \equiv \exists R.\mathtt{Self} & \iff & \mathcal{I} \models \mathtt{id}(C) \equiv \mathtt{id}(\top) \sqcap R \\
\mathcal{I} \models C \sqsubseteq \forall R.D & \iff & \mathcal{I} \models R_{C|} \sqsubseteq R_{|D} \\
\mathcal{I} \models \exists R.C \sqsubseteq D & \iff & \mathcal{I} \models R_{|C} \sqsubseteq R_{D|} \\
\mathcal{I} \models \mathtt{Trans}(R) & \iff & \mathcal{I} \models R \circ R \subseteq R \\
\mathcal{I} \models \mathtt{Irr}(R) & \iff & \mathcal{I} \models R \sqcap \mathtt{id}(\top) = \neg\mathbb{U} \\
\mathcal{I} \models \mathtt{ASym}(R) & \iff & \mathcal{I} \models R \sqcap R^- = \neg\mathbb{U} \\
\mathcal{I} \models C(a) & \iff & \mathcal{I} \models \{a\} \sqsubseteq C \\
\mathcal{I} \models R(a,b) & \iff & \mathcal{I} \models \{a\} \sqsubseteq \exists R.b.
\end{array}
$$

Next, we define a mapping τ from $\mathcal{DL}\langle\forall^\pi\rangle$-statements to $\forall^\pi_{\mathbf{0,2}}$-formulae as follows:

$$
\begin{array}{rcl}
\tau(A \equiv \top) & =_{\text{Def}} & (\forall x \in \Delta)\,(x \in A) \\
\tau(A \equiv \neg B) & =_{\text{Def}} & (\forall x \in \Delta)\,(x \in A \leftrightarrow x \notin B) \\
\tau(A \equiv B \sqcup B') & =_{\text{Def}} & (\forall x \in \Delta)\,(x \in A \leftrightarrow x \in B \vee x \in B') \\
\tau(A \equiv \{a\}) & =_{\text{Def}} & (\forall x \in \Delta)\,(x \in A \leftrightarrow x = a) \wedge a \in A \\
\tau(A \equiv \exists P.\{a\}) & =_{\text{Def}} & (\forall x \in \Delta)\,(x \in A \leftrightarrow [x,a] \in P) \\
\tau(P \equiv \mathbb{U}) & =_{\text{Def}} & (\forall x,y \in \Delta)\,([x,y] \in P) \\
\tau(P \equiv \neg Q) & =_{\text{Def}} & (\forall x,y \in \Delta)\,([x,y] \in P \leftrightarrow [x,y] \notin Q) \\
\tau(P \equiv Q \sqcup Q') & =_{\text{Def}} & (\forall x,y \in \Delta)\,([x,y] \in P \leftrightarrow [x,y] \in Q \vee [x,y] \in Q') \\
\tau(P \equiv Q^-) & =_{\text{Def}} & (\forall x,y \in \Delta)\,([x,y] \in P \leftrightarrow [y,x] \in Q) \\
\tau(P \equiv Q_{A|}) & =_{\text{Def}} & (\forall x,y \in \Delta)\,([x,y] \in P \leftrightarrow [x,y] \in Q \wedge x \in A) \\
\tau(P \equiv \mathtt{id}(A)) & =_{\text{Def}} & (\forall x,y \in \Delta)\,([x,y] \in P \leftrightarrow x = y \wedge x \in A) \\
\tau(P \circ P' \sqsubseteq Q) & =_{\text{Def}} & (\forall [x,y] \in P)\,(\forall [y',z] \in P')\,(y = y' \rightarrow [x,z] \in Q) \\
\tau(\mathtt{Ref}(P)) & =_{\text{Def}} & (\forall [x,y] \in P)\,([x,x] \in P)\,,
\end{array}
$$

for all $A, B, B' \in \mathcal{N}^c$, $P, P', Q \in \mathcal{N}^r$, and $a \in \mathcal{N}^i$. In the previous definition we abbreviated $(\forall x \in \Delta)(\forall y \in \Delta)$ with $(\forall x, y \in \Delta)$.

We remark that in the above definition of the mapping τ we are assuming that the collections $Vars_s$, $Vars_m$ of the variables of the language $\forall^\pi_{\mathbf{0,2}}$ contain all the concept, role, and individual names. In particular $\mathcal{N}^c \cup \mathcal{N}^i \subseteq Vars_s$ and $\mathcal{N}^r \subseteq Vars_m$. Moreover, we used the same symbol Δ which is also

used to denote the domain of a description logic interpretation, under the assumption that $\Delta \in \mathit{Vars}_s \setminus (\mathcal{N}^c \cup \mathcal{N}^r)$. These are just technical assumptions (not strictly necessary for the proof) which have been just introduced to enhance readability of the formulae $\tau(\cdot)$ and to emphasize the strong correlation between the semantical definitions of $\mathcal{DL}\langle\forall^\pi\rangle$-statements and their corresponding $\forall^\pi_{\mathbf{0,2}}$-formulae.

Now let $\mathcal{K}$ be a $\mathcal{DL}\langle\forall^\pi\rangle$-knowledge base. We define the $\forall^\pi_{\mathbf{0,2}}$-formula φ, expressing the consistency of $\mathcal{K}$, as follows

$$\begin{array}{lll}
\varphi & =_{\text{Def}} & \varphi_C \wedge \varphi_R \wedge \varphi_I \wedge \varphi_{\mathcal{K}} \\
\varphi_C & =_{\text{Def}} & \bigwedge\limits_{A \in \mathsf{Cpts}} (\forall x \in A)(x \in \Delta) \\
\varphi_R & =_{\text{Def}} & \bigwedge\limits_{P \in \mathsf{Rls}} (\forall [x,y] \in P)(x \in \Delta \wedge y \in \Delta) \\
\varphi_I & =_{\text{Def}} & \bigwedge\limits_{a \in \mathsf{Inds}} a \in \Delta \\
\varphi_{\mathcal{K}} & =_{\text{Def}} & \bigwedge\limits_{\gamma \in \mathcal{K}} \tau(\gamma),
\end{array}$$

where $\mathsf{Cpts}, \mathsf{Rls}$, and Inds are respectively the sets of concept, role and individual names occurring in $\mathcal{K}$.

The consistency problem for $\mathcal{K}$ is equivalent to the satisfiability of φ, as we prove next.

Plainly, φ_C, φ_R, and φ_I guarantee that each model of φ can easily be turned into a descriptive interpretation, since a model of a $\forall^\pi_{\mathbf{0,2}}$-formula must be map-safe. Additionally, $\varphi_{\mathcal{K}}$ ensures that the $\mathcal{DL}\langle\forall^\pi\rangle$-interpretation obtained in this way satisfies all the statements in $\mathcal{K}$.

Conversely, let $\mathcal{I}$ be a model for $\mathcal{K}$. Without loss of generality, we may assume that $\Delta^{\mathcal{I}}$ is a set belonging to the von Neumann hierarchy $\mathcal{V}$ (otherwise, we embed $\Delta^{\mathcal{I}}$ in $\mathcal{V}$). Thus let $\mathbf{I}$ be any set-theoretic interpretation such that

$$\begin{array}{rll}
M_{\mathbf{I}}\Delta & =_{\text{Def}} \Delta^{\mathcal{I}} & \\
M_{\mathbf{I}}A & =_{\text{Def}} A^{\mathcal{I}} & \text{for all } A \in \mathcal{N}^c \\
M_{\mathbf{I}}P & =_{\text{Def}} P^{\mathcal{I}} & \text{for all } P \in \mathcal{N}^r \\
M_{\mathbf{I}}a & =_{\text{Def}} a^{\mathcal{I}} & \text{for all } a \in \mathcal{N}^i.
\end{array}$$

We remark that we don't need to specify the pairing function when defining a candidate model for a $\forall^\pi_{\mathbf{0,2}}$-formula, in force of the properties of map-isomorphic interpretations reported in Section 4.2.1. In addition, the description logics descriptive semantics guarantees that such any interpretation $\mathbf{I}$ is map-safe, and that $\mathbf{I} \models \varphi$.

We conclude the proof by observing that each conjunct in φ contains at most two quantifiers (i.e., φ is a formula of $(\forall^\pi_{0,2})^{\leq 2}$). Thus, in view of Corollary 12, the satisfiability of φ can be checked in nondeterministic polynomial time, while the NP-hardness of this problem follows directly from the NP-completeness of the satisfiability problem for propositional formulae. □

The reduction provided in Theorem 27 can be easily extended to cope with $\mathcal{DL}\langle\forall^{\pi}\rangle$-knowledge bases extended with finite sets of SWRL rules, which was described in Section 3.5, as shown in the following theorem.

Theorem 28. *The consistency problem for $\mathcal{DL}\langle\forall^{\pi}\rangle$-knowledge bases extended with finite sets of SWRL rules is decidable.*

Proof. Let $\mathcal{K}$ be a $\mathcal{DL}\langle\forall^{\pi}\rangle$-knowledge base, and let $\mathcal{R}$ be a finite set of SWRL rules. Let us extend the mapping τ, defined in Theorem 27, to SWRL rules and atoms as follows:

$$\begin{aligned}
\tau(H \rightarrow B_1 \wedge \ldots \wedge B_n) &=_{\text{Def}} (\forall x_1, \ldots, x_m \in \Delta)\,(\tau H \rightarrow \tau B_1 \wedge \ldots \wedge \tau B_n) \\
\tau(A(x)) &=_{\text{Def}} x \in A \\
\tau(P(x,y)) &=_{\text{Def}} [x,y] \in P \\
\tau(x = y) &=_{\text{Def}} x = y \\
\tau(x \neq y) &=_{\text{Def}} x \neq y,
\end{aligned}$$

where $H, B_1, \ldots B_n$ are SWRL atoms, $x_1, \ldots, x_m$ are the SWRL variables occurring in $H \rightarrow B_1 \wedge \ldots \wedge B_n$, x, y can be either SWRL variables or individual names, and A, P are respectively a concept and a role name. We conclude the proof by observing that the following $\forall^{\pi}_{\mathbf{0,2}}$-formula φ' is satisfiable if and only if the knowledge base $\mathcal{K}$ extended with $\mathcal{R}$ is consistent:

$$\varphi' =_{\text{Def}} \bigwedge_{\rho \in \mathcal{R}} \tau(\rho) \wedge \varphi,$$

where φ is built from $\mathcal{K}$ as described in Theorem 27, extending Cpts, Rls and Inds with the concept, role and individual names occurring in $\mathcal{R}$, respectively. □

The description logic presented in this section can be extended with metamodeling capabilities, in order to allow to represent roles which connect concepts and other roles, and concepts which contains other concepts and roles.

5.3 A Metamodeling Enabled Version of $\mathcal{DL}\langle\forall^{\pi}\rangle$

In this section we introduce $\mathcal{DL}\langle\forall^{\pi}_{\boldsymbol{\Delta}}\rangle$, a metamodeling enabled extension of $\mathcal{DL}\langle\forall^{\pi}\rangle$. We will prove that the consistency problem of $\mathcal{DL}\langle\forall^{\pi}_{\boldsymbol{\Delta}}\rangle$-knowledge bases can be reduced to the satisfiability problem of $(\forall^{\pi}_{\Delta})^{\leq 3}$-formulae. Again, in order to make our proofs more fluent, we will represent $\forall^{\pi}_{\boldsymbol{\Delta}}$-formulae, which are finite conjunctions of prenex $\forall^{\pi}_{\boldsymbol{\Delta}}$-formulae, as finite sets of prenex $\forall^{\pi}_{\boldsymbol{\Delta}}$-formulae. Furthermore, we will assume that $Vars = \mathcal{N}^c \cup \mathcal{N}^r \cup \mathcal{N}^i$.

The description logic $\mathcal{DL}\langle\forall^{\pi}_{\boldsymbol{\Delta}}\rangle$ extends $\mathcal{DL}\langle\forall^{\pi}\rangle$ with the two additional concept constructors $\{R\}$ and $\exists R.\{Q\}$, where R and Q are role terms. Moreover, $\mathcal{DL}\langle\forall^{\pi}_{\boldsymbol{\Delta}}\rangle$ admits also knowledge-base constraints of the forms $C(T)$ and

$R(T, T')$, where C is a $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$-concept, R is a $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$-role, and T, T' are two generic $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$-terms. This allows one to state, for example, that a concept or a role are *deprecated*, i.e. that they belongs to a concept *Deprecated*

$$\mathit{Deprecated}(\mathit{Human}), \quad \mathit{Deprecated}(\mathit{father}),$$

and to indicate which concept and role must be used instead

$$\mathit{replacedBy}(\mathit{Human}, \mathit{HumanBeing}), \quad \mathit{replacedBy}(\mathit{father}, \mathit{isFatherOf}).$$

Formally speaking, $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$-terms are constructed by means of the following syntax rules

$$\begin{array}{rcl} C, D & \longrightarrow & A \mid \top \mid \bot \mid \neg C \mid C \sqcup D \mid C \sqcap D \mid \{a\} \mid \{\mathbf{C}\} \mid \{\mathbf{R}\} \mid \exists R.\mathtt{Self} \mid \\ & & \exists R.\{a\} \mid \exists \mathbf{R}.\{\mathbf{C}\} \mid \exists \mathbf{R}.\{\mathbf{S}\} \\ R, S & \longrightarrow & P \mid \mathbb{U} \mid R^{-} \mid \neg R \mid R \sqcup S \mid R \sqcap S \mid R_{C|} \mid R_{|D} \mid R_{C|D} \mid \\ & & \mathtt{id}(C) \mid \mathtt{sym}(R), \end{array}$$

where A and P are respectively a concept and a role name, C, D are concept terms, R, S are role terms, and a, b are individual names.

$\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$-knowledge bases are finite sets of constraints of the types

$$\begin{array}{llll} C \equiv D, & C \sqsubseteq D, & R \equiv S, & R \sqsubseteq S, \\ C \sqsubseteq \forall R.D, & \exists R.C \sqsubseteq D, & R \circ R' \sqsubseteq S, & \mathtt{Trans}(R), \\ \mathtt{Ref}(R), & \mathtt{ASym}(R), & C(T), & R(T, T'), \end{array}$$

where C, D are concept terms, R, S, S' are role terms, and T, T' can be either concept terms, role terms, or individual names.

The semantics of $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$ differs from the standard description logic semantics as it is multilevel, in order to allow nesting concepts and roles. It is given in terms of $\forall^{\pi}_{\mathbf{\Delta}}$-interpretations as follows. $\forall^{\pi}_{\mathbf{\Delta}}$-interpretations are first extended to complex $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$-terms and to $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$-knowledge base constraints in the usual manner for description logics. Then we say that a $\forall^{\pi}_{\mathbf{\Delta}}$-interpretation $\mathbf{I} = (\Delta^I, M_{\mathbf{I}}, \pi_{\mathbf{I}})$ is a $\forall^{\pi}_{\mathbf{\Delta}}$-model for a $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$-knowledge base $\mathcal{K}$ if and only if it satisfies the following conditions:

(a) $M_{\mathbf{I}}A \subseteq \Delta^I$, for all $A \in \mathcal{N}^c$;

(b) $M_{\mathbf{I}}P \subseteq \Delta^I \times \Delta^I$, for all $P \in \mathcal{N}^r$, where $\times$ is the Cartesian product with respect to $\pi_{\mathbf{I}}$;

(c) $\mathbf{I} \models \alpha$, for each constraint γ occurring in $\mathcal{K}$.

Finally, a $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$-knowledge base is said to be $\forall^{\pi}_{\mathbf{\Delta}}$-consistent if it admits a $\forall^{\pi}_{\mathbf{\Delta}}$-model.

The following lemma shows that $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$ extends $\mathcal{DL}\langle\forall^{\pi}\rangle$ also from a semantic point of view.

Lemma 29. *Let $\mathcal{K}$ be a $\mathcal{DL}\langle\forall^{\pi}\rangle$-knowledge base. $\mathcal{K}$ is consistent (with respect to the descriptive semantics) if and only if it is $\forall^{\pi}_{\mathbf{\Delta}}$-consistent.*

Proof. To begin with, we observe that each $\forall^{\pi}_{\mathbf{\Delta}}$-interpretation $\mathbf{I}$ can be easily turned into a description logic interpretation $\mathcal{I}$ such that $\Delta^{\mathcal{I}} = \Delta^{I}$ and $X^{\mathcal{I}} = M_{\mathbf{I}}X$, for all $X \in \mathit{Vars}$, in such a way that $\mathbf{I}$ is a $\forall^{\pi}_{\mathbf{\Delta}}$-model for $\mathcal{K}$ if and only if $\mathcal{I}$ is a model for $\mathcal{K}$.

For the converse direction, let $\mathcal{I}$ be a model for $\mathcal{K}$. Without loss of generality, we may assume that $\Delta^{\mathcal{I}}$ is a set belonging to the von Neumann hierarchy $\mathcal{V}$ (otherwise, we embed $\Delta^{\mathcal{I}}$ in $\mathcal{V}$). Now, let $\mathbf{I}$ be the $\forall^{\pi}_{\mathbf{\Delta}}$-interpretation defined by

$$\begin{array}{rcl} \Delta^{I} & =_{\text{Def}} & \Delta^{\mathcal{I}} \cup \left\{X^{\mathcal{I}} : X \in \mathcal{N}^{c} \cup \mathcal{N}^{r}\right\} \\ \pi_{\mathbf{I}}(u,v) & =_{\text{Def}} & \{\{u\},\{u,v\},\Delta^{I}\} \\ M_{\mathbf{I}}P & =_{\text{Def}} & \{\pi_{\mathbf{I}}(u,v) : [u,v] \in P^{\mathcal{I}}\} \\ M_{\mathbf{I}}Y & =_{\text{Def}} & Y^{\mathcal{I}}, \end{array}$$

for all $u, v \in \mathcal{V}, P \in \mathcal{N}^{r}, Y \in \mathcal{N}^{c} \cup \mathcal{N}^{i}$.

This definition guarantees that $\mathbf{I}$ satisfies *(a)* and *(b)*. Furthermore, it can be proved by induction on the terms structure that $\mathbf{I} \models \alpha$ if and only if $\mathcal{I} \models \alpha$, for each possible $\mathcal{DL}\langle\forall^{\pi}\rangle$-knowledge base constraint α, and thus $\mathbf{I}$ must satisfy *(c)* also, since $\mathcal{I}$ is a model for $\mathcal{K}$. □

As mentioned before, the language $(\forall^{\pi}_{\Delta})^{\leq 3}$ is expressive enough to include $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$. In fact, from any $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$-knowledge base $\mathcal{K}$ we can construct an *equivalent* $(\forall^{\pi}_{\Delta})^{\leq 3}$-formula φ as indicated in the following theorem.

Theorem 30. *The consistency problem for $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$-knowledge bases is* NP*-complete.*

Proof. Let $\mathcal{K}$ be a $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$-knowledge base. We show how to construct an equivalent $(\forall^{\pi}_{\Delta})^{\leq 3}$-formula φ. We assume, without loss of generality, that $\mathcal{K}$ consists only of constraints of the forms indicated in the left column of Table 5.2. φ is defined as follows:

$$\varphi =_{\text{Def}} \mathcal{K}_0 \cup \bigcup_{\gamma \in \mathcal{K}} \rho(\gamma),$$

where

$$\begin{array}{rcl} \mathcal{K}_0 =_{\text{Def}} & \displaystyle\bigcup_{A \in \mathcal{N}^{c} \cap \mathit{Vars}(\mathcal{K})} & \{(\forall x)(\forall y)([x,y] \notin A)\} \\ & \cup \displaystyle\bigcup_{P \in \mathcal{N}^{r} \cap \mathit{Vars}(\mathcal{K})} & \{(\forall x)(x \notin P)\} \end{array}$$

and ρ is the transformation defined in Table 5.2.

It can easily be verified that, in force of the constraints in $\mathcal{K}_0$, each model of φ must satisfies *(a)* and *(b)*. Furthermore, given any $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$-interpretation $\mathbf{I}$, $\mathbf{I}$ evaluates γ to **true** if and only if it is a model for $\rho(\gamma)$, for any constraint γ occurring in $\mathcal{K}$.

γ	$\rho(\gamma)$
$A \equiv \top$	$\{(\forall x)(x \in A)\}$
$A \equiv \neg B$	$\{(\forall x)(x \in A \leftrightarrow x \notin A)\}$
$A \equiv B \sqcup B'$	$\{(\forall x)(x \in A \leftrightarrow x \in B \vee x \in B')\}$
$A \equiv \{X\}$	$\{X \in A; (\forall x)(x \in A \rightarrow x = X)\}$
$A \sqsubseteq \forall P.B$	$\{(\forall x)(\forall y)(x \in A \wedge [x,y] \in P \rightarrow y \in B)\}$
$\exists P.A \sqsubseteq B$	$\{(\forall x)(\forall y)(y \in A \wedge [x,y] \in P \rightarrow x \in B)\}$
$A \equiv \exists P.\{X\}$	$\{(\forall x)(x \in A \leftrightarrow [x,X] \in P)\}$
$P \equiv \mathbb{U}$	$\{(\forall x)(\forall y)([x,y] \in P)\}$
$P \equiv \neg Q$	$\{(\forall x)(\forall y)([x,y] \in P \leftrightarrow [x,y] \notin Q)\}$
$P \equiv Q \sqcup Q'$	$\{(\forall x)(\forall y)([x,y] \in P \leftrightarrow ([x,y] \in Q \vee [x,y] \in Q'))\}$
$P \equiv Q^-$	$\{(\forall x)(\forall y)([x,y] \in P \leftrightarrow [y,x] \in Q)\}$
$P \equiv \mathtt{id}(A)$	$\{(\forall x)(\forall y)([x,y] \in P \leftrightarrow x = y \wedge x \in A)\}$
$P \equiv Q_{A\vert}$	$\{(\forall x)(\forall y)([x,y] \in P \leftrightarrow ([x,y] \in Q \wedge x \in A))\}$
$P \circ P' \sqsubseteq Q$	$\{(\forall x)(\forall y)(\forall z)(([x,y] \in P \wedge [y,z] \in P') \rightarrow [x,z] \in Q)\}$
$\mathtt{Ref}(P)$	$\{(\forall x)(\forall y)([x,y] \in P \rightarrow [x,x] \in P)\}$

Table 5.2: *The transformation ρ.*

In addition, this construction process takes polynomial time, with respect to the size of $\mathcal{K}$, and thus the consistency problem for $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$-knowledge bases can be tested in nondeterministic polynomial time. The NP-hardness of this problem follows from the NP-completeness of reasoning for $\mathcal{DL}\langle\forall^{\pi}\rangle$ (cf. Corollary 23). □

We conclude this section by observing that $\forall^{\pi}_{\mathbf{\Delta}}$ (and also its restriction $(\forall^{\pi}_{\Delta})^{\leq 3}$) is much more flexible than the description logic introduced here, and thus it may be used to extend $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$ with new constructs to increase its metamodeling capabilities. We clarify this observation by defining, for instance, a novel construct involving the union set operator:

$$\bigcup A \sqsubseteq B \quad =_{\text{Def}} \quad (\forall x)(\forall y)((x \in y \wedge y \in A) \rightarrow z \in B)\,,$$

where $A, B \in \mathcal{N}^c$.

Chapter 6

Conclusions and Future Work

In this thesis we presented three quantified fragments of set theory, namely $\forall^{\pi}_{\mathbf{0}}$, $\forall^{\pi}_{\mathbf{0,2}}$, and $\forall^{\pi}_{\mathbf{\Delta}}$. In $\forall^{\pi}_{\mathbf{0}}$ only restricted quantifiers of the forms $(\forall x \in \bar{\pi}(y))$, $(\forall [x, y] \in z)$ are permitted. $\forall^{\pi}_{\mathbf{0,2}}$ is a two sorted language which allows quantifiers of the forms $(\forall x \in y)$ and $(\forall [x, y] \in f)$, with $x, y \in \mathit{Vars}_s$, $f \in \mathit{Vars}_m$. In $\forall^{\pi}_{\mathbf{\Delta}}$ quantifiers are not restricted, rather quantified variables ranges over a set Δ, and interpretations must satisfies the additional constraint $\Delta \cap (\Delta \times \Delta) = \emptyset$ to be models for $\forall^{\pi}_{\mathbf{\Delta}}$-formulae.

We proved that all these languages, which we call $\forall^{\pi}$-languages, are decidable, and that their decision problem is in NExpTime. In addition, if we restrict ourselves to formulae with quantifier prefixes whose length is bounded by a constant, the relative decision problem becomes NP-complete.

Next, we showed that a large amount of set-theoretic constructs can be expressed by these languages. To this purpose, we presented the unquantified language $\mathsf{MLSS}^{\times}_{2,m}$, which contains the usual set-theoretic operators $\cup$, $\cap$, $\setminus$, and several map constructs like, for example, map restrictions and Cartesian product.

We exploited some correspondences between set-theoretical languages and description logics. In particular, we proved that each MLS extension which contains the map image operator is ExpTime-hard by providing a reduction from the description logic $\mathcal{ALC}$ to the MLS extension MLS_{Im}.

Conversely, several description logic constructs are $\forall^{\pi}$-*expressible*, proving that the languages of this family may be profitably employed as knowledge representation languages. In particular, the decision problem of the description logic $\mathcal{DL}\langle\forall^{\pi}\rangle$ can be reduced in a very natural way to the decision problem of $(\forall^{\pi}_{0,2})^{\leq 2}$, thus proving that reasoning in $\mathcal{DL}\langle\forall^{\pi}\rangle$ is NP-complete. In addition, we showed that this description logic can be extended with SWRL rules without disrupting the decidability. Finally, we introduced the description logic $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$, which extends $\mathcal{DL}\langle\forall^{\pi}\rangle$ with metamodeling features, and we proved that reasoning in $\mathcal{DL}\langle\forall^{\pi}_{\mathbf{\Delta}}\rangle$ is still NP-complete by providing a reduction to $(\forall^{\pi}_{\Delta})^{\leq 3}$.

We intend to further explore quantified fragments of set theory, and their applicability in knowledge representation, thus focusing our attention on

fragments of set theory with a *tractable* decision problem (in NP), and which contain map-related (or pair-related) constructs. For this reason, we intend to investigate the decision problem for the one-sorted language obtained from $\forall^{\pi}_{\mathbf{0},\mathbf{2}}$ by dropping the distinction between set and map variables. We conjecture that an undecidability result for this theory can be proved by a reduction of the tiling problem (cf. [4]).

The language $\mathsf{MLSS}^{\times}_{2,m}$ imposes some relevant restrictions on the usage of the domain and range operators. However, these restrictions can be relaxed when these operators are applied to reflexive functions. In light of this, $\mathsf{MLSS}^{\times}_{2,m}$ extensions which allow one to use the domain operator also on the right-hand side of inclusions should be investigated, when the application of such kind of constructs is limited, for example, to symmetric or transitive functions.

Also in $\mathcal{DL}\langle\forall^{\pi}\rangle$ some very common description logic operators are available in a limited fashion, in particular the universal and the existential restrictions. In [14], the knowledge bases in the test bed provided in [20] were examined in order to estimate if and how much the description logic $\mathcal{DL}\langle\mathsf{MLSS}^{\times}_{2,m}\rangle$ (see [13]) can cope with real-world knowledge bases. However, this test bed is out-of-date, and therefore it should be repeated with a most relevant one, relatively to the description logic $\mathcal{DL}\langle\forall^{\pi}\rangle$, which strictly contains $\mathcal{DL}\langle\mathsf{MLSS}^{\times}_{2,m}\rangle$.

In [29, 27] some correspondences between description logics and *Disjunctive Datalog* (cf. [18]) have been exploited. In particular, in [29] a description logic which can be reduced to Datalog was devised, so that all the highly optimized reasoning engines for Datalog developed over the years may be reused in this context. For this reason, the relationship between our languages and Datalog should be investigated.

Bibliography

[1] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 364–369. Professional Book Center, 2005.

[2] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[3] José L. Balcázar. The Complexity of Searching Implicit Graphs. *Artif. Intell.*, 86(1):171–188, 1996.

[4] R Berger. *The undecidability of the dominoe problem.*, volume 66 of *Mem. Amer. Math. Soc.*, pages 1–72. American Mathematical Society, 1966.

[5] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American Magazine*, May 2001.

[6] Michael Breban, Alfredo Ferro, Eugenio G. Omodeo, and Jacob T. Schwartz. Decision procedures for elementary sublanguages of set theory. II. Formulas involving restricted quantifiers, together with ordinal, integer, map, and domain notions. *Communications on Pure and Applied Mathematics*, 34:177–195, 1981.

[7] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *dl-lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[8] D. Cantone, E. G. Omodeo, and A. Policriti. The automation of syllogistic. ii. optimization and complexity issues. *J. Autom. Reason.*, 6:173–187, May 1990.

[9] Domenico Cantone, Vincenzo Cutello, and Alberto Policriti. Set-theoretic reductions of hilbert's tenth problem. In Egon Börger, Hans Kleine Büning, and Michael M. Richter, editors, *CSL*, volume 440 of *Lecture Notes in Computer Science*, pages 65–75. Springer, 1989.

[10] Domenico Cantone and Alfredo Ferro. *Techniques of computable set theory with applications to proof verification*, volume XLVIII of *Comm. Pure Appl. Math.*, pages 901–945. Wiley, 1995.

[11] Domenico Cantone, Alfredo Ferro, and Eugenio Omodeo. *Computable set theory*, volume 6 of *International Series of Monographs on Computer Science*. Oxford Science Publications. Clarendon Press, Oxford, UK, 1989.

[12] Domenico Cantone, Cristiano Longo, and Marianna Nicolosi Asmundo. A Decidable Quantified Fragment of Set Theory Involving Ordered Pairs with Applications to Description Logics. In Marc Bezem, editor, *CSL*, volume 12 of *LIPIcs*, pages 129–143. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[13] Domenico Cantone, Cristiano Longo, and Marianna Nicolosi Asmundo. A Decision Procedure for a Two-sorted Extension of Multi-Level Syllogistic with the Cartesian Product and Some Map Constructs. In Wolfgang Faber and Nicola Leone, editors, *CILC2010 : 25th Italian Conference on Computational Logic*, 2010.

[14] Domenico Cantone, Cristiano Longo, and Antonio Pisasale. Comparing Description Logics with Multi-level Syllogistics: the Description Logic $\mathcal{DL}\langle\mathsf{MLSS}^{\times}_{2,m}\rangle$. In *6th Workshop on Semantic Web Applications and Perspectives (SWAP)*, 2010.

[15] Domenico Cantone, Eugenio Omodeo, and Alberto Policriti. *Set theory for computing: from decision procedures to declarative programming with sets*. Monographs in Computer Science. Springer-Verlag, New York, NY, USA, 2001.

[16] Domenico Cantone, Eugenio G. Omodeo, Jacob T. Schwartz, and Pietro Ursino. Notes from the logbook of a proof-checker's project. In Nachum Dershowitz, editor, *Verification: Theory and Practice*, volume 2772 of *Lecture Notes in Computer Science*, pages 182–207. Springer, 2003.

[17] Domenico Cantone and Jacob T. Schwartz. Decision Procedures for Elementary Sublanguages of Set Theory: XI. Multilevel Syllogistic Extended by Some Elementary Map Constructs. *J. Autom. Reasoning*, 7(2):231–256, 1991.

[18] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive Datalog. *ACM Trans. Database Syst.*, 22(3):364–418, 1997.

[19] Alfredo Ferro, Eugenio G. Omodeo, and Jacob T. Schwartz. *Decision Procedures for Elementary Sublanguages of Set Theory. I. Multi-level syllogistic and some extensions.*, volume XXXIII of *Comm. Pure Appl. Math.*, pages 599–608. Wiley, 1980.

[20] Tom Gardiner, Dmitry Tsarkov, and Ian Horrocks. Framework for an automated comparison of description logic reasoners. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 654–667. Springer, 2006.

[21] Birte Glimm, Sebastian Rudolph, and Johanna Völker. Integrated metamodeling and diagnosis in owl 2. In Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang 0007, Jeff Z. Pan, Ian Horrocks, and Birte Glimm, editors, *International Semantic Web Conference (1)*, volume 6496 of *Lecture Notes in Computer Science*, pages 257–272. Springer, 2010.

[22] Volker Haarslev and Ralf Möller. RACE System Description. In Lambrix et al. [30].

[23] Ian Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In *KR*, pages 636–649, 1998.

[24] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The Even More Irresistible SROIQ. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006)*, pages 57–67. 10th International Conference on Principles of Knowledge Representation and Reasoning, AAAI Press, June 2006.

[25] Ian Horrocks and Peter F. Patel-Schneider. Reducing OWL Entailment to Description Logic Satisfability. In Diego Calvanese, Giuseppe De Giacomo, and Enrico Franconi, editors, *Description Logics*, volume 81 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.

[26] Ian Horrocks and Peter F. Patel-Schneider. A proposal for an OWL rules language. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *WWW*, pages 723–731. ACM, 2004.

[27] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reducing SHIQ-Description Logic to Disjunctive Datalog Programs. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *KR*, pages 152–162. AAAI Press, 2004.

[28] Markus Krötzsch. *Description Logic Rules*, volume 008 of *Studies on the Semantic Web*. IOS Press/AKA, 2010.

[29] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Description Logic Rules. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikolaos M. Avouris, editors, *ECAI*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 80–84. IOS Press, 2008.

[30] Patrick Lambrix, Alexander Borgida, Maurizio Lenzerini, Ralf Möller, and Peter F. Patel-Schneider, editors. *Proceedings of the 1999 International Workshop on Description Logics (DL'99), Linköping, Sweden, July 30 - August 1, 1999*, volume 22 of *CEUR Workshop Proceedings*. CEUR-WS.org, 1999.

[31] Boris Motik. On the Properties of Metamodeling in OWL. *J. Log. Comput.*, 17(4):617–637, 2007.

[32] Boris Motik, Ulrike Sattler, and Rudi Studer. Query Answering for OWL-DL with rules. *J. Web Sem.*, 3(1):41–60, 2005.

[33] Eugenio Omodeo, Domenico Cantone, and Alberto Policriti. Reasoning, Action and Interaction in AI Theories and Systems, Essays Dedicated to Luigia Carlucci Aiello. In Oliviero Stock and Marco Schaerf, editors, *Reasoning, Action and Interaction in AI Theories and Systems*, volume 4155 of *Lecture Notes in Computer Science*. Springer, 2006.

[34] Jeff Z. Pan and Ian Horrocks. OWL FA: a metamodeling extension of OWL DL. In Les Carr, David De Roure, Arun Iyengar, Carole A. Goble, and Michael Dahlin, editors, *WWW*, pages 1065–1066. ACM, 2006.

[35] Jeff Z. Pan and Ian Horrocks. RDFS(FA): Connecting RDF(S) and OWL DL. *IEEE Trans. Knowl. Data Eng.*, 19(2):192–206, 2007.

[36] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[37] Franco Parlamento and Alberto Policriti. Undecidability results for restricted universally quantified formulae of set theory. *Comm. Pure Appl. Math.*, ILVI:57–73, 1993.

[38] Franco Parlamento, Alberto Policriti, and K. P. S. B. Rao. Witnessing differences without redundancies. In *Proc. Amer. Math. Soc.*, volume 125, pages 587–594, 1997.

[39] Peter F. Patel-Schneider. DLP. In Lambrix et al. [30].

[40] M. Ross Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing*, pages 216–270. The MIT Press, 1968.

[41] Jacob T. Schwartz. A survey of program proof technology. Technical Report 001, New York University, Department of Computer Science, September 1978.

[42] Jacob T. Schwartz, Domenico Cantone, and Eugenio G. Omodeo. *Computational Logic and Set Theory: Applying Formalized Logic to Analysis*. Texts in Computer Science. Springer-Verlag New York, Inc., 2011.

[43] Jacob T. Schwartz, Robert B. K. Dewar, Edmond Schonberg, and E Dubinsky. *Programming with sets; an introduction to SETL*. Springer-Verlag New York, Inc., New York, NY, USA, 1986.

[44] Stephan Tobies. The Complexity of Reasoning with Cardinality Restrictions and Nominals in Expressive Description Logics. *CoRR*, abs/1106.0239, 2011.

[45] Thanh Tran, Peter Haase, Boris Motik, Bernardo Cuenca Grau, and Ian Horrocks. Metalevel information in ontology-based applications. In Dieter Fox and Carla P. Gomes, editors, *AAAI*, pages 1237–1242. AAAI Press, 2008.

[46] Christopher A. Welty and David A. Ferrucci. What's in an instance? Technical report, RPI Computer Science, 1994.