A Reduction Approach to Decision Procedures

Deepak Kapur and Calogero G. Zarba

University of New Mexico

Abstract. We present an approach for designing decision procedures based on the reduction of complex theories to simpler ones. Specifically, we define *reduction functions* as a tool for reducing the satisfiability problem of a complex theory to the satisfiability problem of a simpler one.
Reduction functions allow us to reduce the theory of lists to the theory of constructors, the theory of arrays to the theory of equality, the theory of sets to the theory of equality, and the theory of multisets to the theory of integers.
Finally, we provide a method for combining reduction functions. This method allows us to reduce the satisfiability problem of a combination of complex theories to the combination of simpler ones.

1 Introduction

In program verification one has often to decide the satisfiability or validity of logical formulae spanning several decidable model classes[1] such as:

- the model class $M_{\approx}$ of equality;
- the model class M_{int} of integers;
- the model class M_{cons} of constructors;
- the model class M_{list} of lists;[2]
- the model class M_{array} of arrays;
- the model class M_{set} of sets;
- the model class M_{bag} of multisets.

[1] In the abstract of this paper we have used word “theory”. Most researchers define a (first-order) theory as a set of first-order sentences. According to the researcher's taste, this set may or may not be recursively enumerable, and it may or may not be closed under first-order deductions. When writing this paper, we did not like to work with theories as sets of sentences. We like more to work directly with the models of the theory. For instance, instead of working with the “theory of Presburger arithmetic” by dealing with the set of all sentences that are true in the standard model $(\mathbb{N}, 0, 1, +, \leq)$ of Presburger arithmetic, we prefer to work directly with the standard model $(\mathbb{N}, 0, 1, +, \leq)$. Therefore, in this paper we do not really use theories, but rather we use *model classes*. In standard model-theoretic terminology, a model class is a set of structures closed under isomorphism.

[2] In this paper, M_{list} is an extension of M_{cons}. Specifically M_{cons} is a model class of flat, acyclic, linear lists constructed using the constructors nil and cons. M_{list} is obtained by extending M_{cons} with the selectors car and cdr.

This goal can be achieved by integrating a SAT solver with a decision procedure P for the satisfiability of conjunctions of literals in the combined model class

$$M_0 = M_{\approx} \oplus M_{\mathsf{int}} \oplus M_{\mathsf{list}} \oplus M_{\mathsf{array}} \oplus M_{\mathsf{set}} \oplus M_{\mathsf{bag}} \, .$$

It is often desirable that the decision procedure P be able to return more than just a yes/no answer. For an efficient integration with the SAT solver, P may be required to return implied literals and mininal conflict set of literals. For trusting reasons, P may be required to return a checkable proof that an input conjunction is unsatisfiable. For debugging reasons, P may be required to return a model when an input conjunction is satisfiable.

Due to the complexity of the model class M_0 and the stringent requirements that are asked to P, implementing P is a very daunting task.

To sidestep this difficulty, we propose a different approach based on the reduction from complex model classes to simpler model classes. Instead of implementing a decision procedure for M_0, we prefer to implement a decision procedure for the simpler model class

$$N_0 = M_{\approx} \oplus M_{\mathsf{int}} \oplus M_{\mathsf{cons}} \, .$$

Then, we use *reduction functions* in order to reduce any formula φ in the model class M_0 to an equisatisfiable formula ψ in the model class N_0.

Since N_0 is much simpler than M_0, it is easier to implement a decision procedure for N_0 that is able to return implied literals, minimal conflict sets, checkable proofs, and models.

The reduction approach is particularly attractive to us since decision procedures for $M_{\approx}$, M_{int}, and M_{cons} have already been implemented and extensively used in Rewrite Rule Laboratory (RRL) [5]. We have considerable experience in developing heuristics for handling satisfiability problems for such model classes. Heuristics for integrating these model classes into contextual rewriting as well as induction theorem proving based on the cover set method have already been developed [4]. Our recent work on integrating decision procedures with induction for automatically deciding a subclass of inductive conjectures is also based on the model classes M_{int} and M_{cons} [3].

1.1 Contributions

The contribution of this paper are as follows.

1. We introduce the notion of *reduction functions*. Intuitively, a reduction function ρ from a model class M to a model class N translates every formula φ into a formula ψ such that φ is satisfiable in M if and only if ψ is satisfiable in N.
2. We prove that:
   - the model class M_{list} of lists reduces to the model class M_{cons} of constructors;

- the model class M_{array} of arrays reduces to the model class $M_{\approx}$ of equality;
- the model class M_{set} of sets reduces to the model class $M_{\approx}$ of equality;
- the model class M_{bag} of multisets reduces to the model class M_{int} of integers extended with uninterpreted function symbols.

3. We provide a method for *combining reduction functions*. More precisely, assume that ρ_i is a reduction function from M_i to N_i, for $i = 1, 2$, and that the signatures of M_1 and M_2 do not share any function or predicate symbols. Then it is possible to use ρ_1 and ρ_2 as black boxes in order to construct a reduction function ρ from $M_1 \oplus M_2$ to $N_1 \oplus N_2$.

By using the results in contribution 2, and by repeatedly applying the combination result in contribution 3, we are able to implement a reduction function from the complex model class M_0 to the simpler model class N_0.

1.2 Organization of the paper.

In Section 2 we introduce the syntax and semantics of many-sorted logic. In Section 3 we introduce several model classes of interest to program verification. In Section 4 we present a fundamental combination result indipendently due to Ringeissen [8] and Tinelli and Harandi [10]. In Section 5 we introduce normalization functions. In Section 6 we introduce purification functions. In Section 7 we introduce reduction functions. In Section 8 we prove that the model class M_{list} of lists effectively reduces to the model class M_{cons} of constructors. In Section 9 we prove that the model class M_{array} of arrays effectively reduces to the model class $M_{\approx}$ of equality. In Section 10 we prove that the model class M_{set} of sets effectively reduces to the model class $M_{\approx}$ of equality. In Section 11 we prove that the model class M_{bag} of multisets effectively reduces to the model class M_{int} of integers extended with uninterpreted function symbols. In Section 12 we present a method for combining reduction functions. In Section 13 we draw final conclusions.

2 Many-sorted logic

2.1 Syntax

A *signature* Σ is a triple (S, F, P) where S is a set of *sorts*, F is a set of *function symbols*,[3] P is a set of *predicate symbols*, and all the symbols in F, P have arities constructed using the sorts in S.[4] Given a signature $\Sigma = (S, F, P)$, we write Σ^{S} for S, Σ^{F} for F, and Σ^{P} for P. If $\Sigma_1 = (S_1, F_1, P_1)$ and $\Sigma_2 = (S_2, F_2, P_2)$ are signatures, we write $\Sigma_1 \subseteq \Sigma_2$ when $S_1 \subseteq S_2$, $F_1 \subseteq F_2$, and $P_1 \subseteq P_2$. If $\Sigma_1 = (S_1, F_1, P_1)$ and $\Sigma_2 = (S_2, F_2, P_2)$ are signatures, their *union* is the

[3] We regard *constant symbols* as 0-ary function symbols.

[4] According to us, signatures do not specify the arities of function and predicate symbols. In other words, the arity of a symbol is built-in in the symbol itself.

signature $\Sigma_1 \cup \Sigma_2 = (S_1 \cup S_2, F_1 \cup F_2, P_1 \cup P_2)$, and their *intersection* is the signature $\Sigma_1 \cap \Sigma_2 = (S_1 \cap S_2, F_1 \cap F_2, P_1 \cap P_2)$.

For each sort σ, we fix a set $\mathcal{X}_\sigma$ of *free constant symbols* of sort σ. We assume that $\mathcal{X}_\sigma \cap \Sigma^{\mathrm{F}} = \emptyset$, for each signature Σ. We also fix an infinite set $\mathcal{X}_{\mathsf{bool}}$ of *free propositional symbols*. *Free symbols* are either free constant symbols or free propositional symbols.

If X is a set of free symbols and σ is a sort, we denote with X_σ the set of all free constant symbols of sort σ contained in X. Likewise, X_{bool} is the set of all free propositional symbols contained in X.

Definition 1. Let Σ be a signature. The set of Σ-TERMS of sort σ is the smallest set satisfying the following conditions:

- Each free constant symbol $u \in \mathcal{X}_\sigma$ is a Σ-term of sort σ, provided that $\sigma \in \Sigma^{\mathrm{S}}$.
- Each constant symbol $u \in \Sigma^{\mathrm{F}}$ of sort σ is a Σ-term of sort σ.
- $f(t_1, \ldots, t_n)$ is a Σ-term of sort σ, provided that $f \in \Sigma^{\mathrm{F}}$ is a function symbol of arity $\sigma_1 \times \cdots \times \sigma_n \to \sigma$ and t_i is a Σ-term of sort σ_i, for $i = 1, \ldots, n$. $\square$

Definition 2. Let Σ be a signature. The set of Σ-ATOMS is the smallest set satisfying the following conditions:

- Each propositional symbol $u \in \mathcal{X}_{\mathsf{bool}}$ is a Σ-atom;
- $s \approx t$ is a Σ-atom, provided that s, t are Σ-terms of the same sort;[5]
- $p(t_1, \ldots, t_n)$ is a Σ-atom, provided that $p \in \Sigma^{\mathrm{P}}$ is a predicate symbol of arity $\sigma_1 \times \cdots \times \sigma_n$ and t_i is a Σ-term of sort σ_i, for $i = 1, \ldots, n$. $\square$

Definition 3. Let Σ be a signature. The set of Σ-FORMULAE is the smallest set satisfying the following conditions:

- Each Σ-atom is a Σ-formula;
- If φ, ψ, and χ are Σ-formulae, so are $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \to \psi$, $\varphi \leftrightarrow \psi$, and *if* φ *then* ψ *else* χ.[6] $\square$

We drop the prefix 'Σ-' from 'Σ-term', 'Σ-atom', and 'Σ-formula' whenever Σ is irrelevant to the context.

If φ is a term or formula, we denote with *free*(φ) the set of all free constant symbols and free propositional symbols occurring in φ.

In the following, we write $s \approx_\sigma t$ whenever we want to emphasize that s and t are terms of sort σ. We also write $s \not\approx t$ as a shorthand of $\neg(s \approx t)$.

[5] In this paper, the equality symbol $\approx$ is a logical one. Thus, the symbol $\approx$ does not belong to any signature.

[6] Note that in this paper all formulae are quantifier-free.

2.2 Semantics

Definition 4. Let Σ be a signature, and let X be a set of free symbols. Assume that each free constant symbol in X has a sort in Σ^{S}. A Σ-STRUCTURE over X is a map which interprets:

- each sort $\sigma \in \Sigma^{\mathrm{S}}$ as a nonempty domain A_σ;
- each free constant symbols u in X_σ as an element $u^{\mathcal{A}} \in A_\sigma$;
- each free propositional symbol $u \in X_{\mathsf{bool}}$ as a truth value in $\{\mathit{true}, \mathit{false}\}$;
- each function symbol $f \in \Sigma^{\mathrm{F}}$ of arity $\sigma_1 \times \cdots \times \sigma_n \to \sigma$ as a function $f^{\mathcal{A}} : A_{\sigma_1} \times \cdots \times A_{\sigma_n} \to A_\sigma$;
- each predicate symbol $p \in \Sigma^{\mathrm{P}}$ of arity $\sigma_1 \times \cdots \times \sigma_n$ as a subset $p^{\mathcal{A}} \subseteq A_{\sigma_1} \times \cdots \times A_{\sigma_n}$. □

Let $\mathcal{A}$ be a Σ-structure over X, and let φ be either a Σ-term or a Σ-formula such that $\mathit{free}(\varphi) \subseteq X$. We denote with $\varphi^{\mathcal{A}}$ the evaluation of φ under $\mathcal{A}$. Moreover, when φ is a formula, we write $\mathcal{A} \models \varphi$ whenever $\varphi^{\mathcal{A}} = \mathit{true}$.

Definition 5. A Σ-formula φ is SATISFIABLE if $\mathcal{A} \models \varphi$, for some some Σ-structure $\mathcal{A}$ over $\mathit{free}(\varphi)$. □

Let $\mathcal{A}$ be a Σ-structure over X, let x be a free constant of sort σ, and let $a \in A_\sigma$. Assume that $x \notin X$. We denote with $\mathcal{A}\{x/a\}$ the Σ-structure over $X \cup \{a\}$ that extends $\mathcal{A}$ by interpreting the free constant x as a.

Let $\mathcal{A}$ be a Σ-structure over X. Then:

- For $\Sigma_0 \subseteq \Sigma$ and $X_0 \subseteq X$, we denote with $\mathcal{A}^{\Sigma_0, X_0}$ the structure obtained from $\mathcal{A}$ by restricting it to interpret only the symbols in Σ_0 and the free symbols in X_0. Furthermore, we let $\mathcal{A}^{\Sigma_0} = \mathcal{A}^{\Sigma_0, \emptyset}$.
- For $S_0 \subseteq \Sigma^{\mathrm{S}}$ and $X_0 \subseteq X$, we let $\mathcal{A}^{S_0, X_0} = \mathcal{A}^{\Omega, X_0}$, where $\Omega = (S_0, \emptyset, \emptyset)$. Furthermore, we let $\mathcal{A}^{S_0} = \mathcal{A}^{S_0, X_0}$.

Definition 6. Let $\mathcal{A}$ and $\mathcal{B}$ be two Σ-structures over X. An ISOMORPHISM h of $\mathcal{A}$ into $\mathcal{B}$ is a family of bijective functions

$$h = \left\{ h_\sigma : A_\sigma \to B_\sigma \mid \sigma \in \Sigma^{\mathrm{S}} \right\}$$

such that:

- $h_\sigma(u^{\mathcal{A}}) = u^{\mathcal{B}}$, for each free constant symbol $u \in X_\sigma$;
- $u^{\mathcal{A}} = u^{\mathcal{B}}$, for each free propositional symbol $u \in X_{\mathsf{bool}}$;
- $h_\sigma(f^{\mathcal{A}}(a_1, \ldots, a_n)) = f^{\mathcal{B}}(h_{\sigma_1}(a_1), \ldots, h_{\sigma_n}(a_n))$, for each function symbol $f \in \Sigma^{\mathrm{F}}$ of arity $\sigma_1 \times \cdots \times \sigma_n \to \sigma$;
- $(a_1, \ldots, a_n) \in p^{\mathcal{A}}$ if and only if $(h_{\sigma_1}(a_1), \ldots, h_{\sigma_n}(a_n)) \in p^{\mathcal{B}}$, for each predicate symbol $p \in \Sigma^{\mathrm{P}}$ of arity $\sigma_1 \times \cdots \times \sigma_n$. □

We write $\mathcal{A} \cong \mathcal{B}$ when there is an isomorphism of $\mathcal{A}$ into $\mathcal{B}$.

Proposition 7. *Let $\mathcal{A}$ and $\mathcal{B}$ be Σ-structures over X, and let φ be a Σ-formula such that $\mathit{free}(\varphi) \subseteq X$. Assume that $\mathcal{A} \cong \mathcal{B}$. Then*

$$\mathcal{A} \models \varphi \qquad \Longleftrightarrow \qquad \mathcal{B} \models \varphi\,. \qquad \square$$

3 Model Classes

Definition 8. A Σ-MODEL CLASS is a pair $M = (\Sigma, \mathbf{A})$ where

- Σ is a signature;
- $\mathbf{A}$ is a class of Σ-structures over the empty set $\emptyset$;
- $\mathbf{A}$ is closed under isomorphism. □

Definition 9. Let $M = (\Sigma, \mathbf{A})$ be a model class, and let $\mathcal{A}$ be a Σ-structure over X. We say that $\mathcal{A}$ is an M-STRUCTURE if $\mathcal{A}^{\Sigma} \in \mathbf{A}$. □

Let M be a Σ-model class, let φ be a Σ-formula, and let $\mathcal{A}$ be a Σ-structure over $\mathit{free}(\varphi)$. We write $\mathcal{A} \models_M \varphi$ whenever $\varphi^{\mathcal{A}} = \mathit{true}$ and $\mathcal{A}$ is a M-structure.

Definition 10. Given a Σ-model class M, a Σ-formula φ is M-SATISFIABLE if $\mathcal{A} \models_M \varphi$, for some Σ-structure $\mathcal{A}$ over $\mathit{free}(\varphi)$. □

Definition 11. Given a Σ-model class M, the SATISFIABILITY PROBLEM of M is the problem of deciding, for each Σ-formula φ, whether or not φ is M-satisfiable. □

Proposition 12. *Let M be a Σ-model class, let $\mathcal{A}$ and $\mathcal{B}$ be Σ-structures over X, and let φ be a Σ-formula such that $\mathit{free}(\varphi) \subseteq X$. Assume that $\mathcal{A} \cong \mathcal{B}$. Then*

$$\mathcal{A} \models_M \varphi \qquad \Longleftrightarrow \qquad \mathcal{B} \models_M \varphi\,. \qquad \square$$

Definition 13. Let $M_i = (\Sigma_i, \mathbf{A}_i)$ be a model class, for $i = 1, 2$. The COMBINATION of M_1 and M_2 is the model class $M_1 \oplus M_2 = (\Sigma, \mathbf{A})$ where $\Sigma = \Sigma_1 \cup \Sigma_2$ and $\mathbf{A} = \{\mathcal{A} \mid \mathcal{A}^{\Sigma_1} \in \mathbf{A}_1 \text{ and } \mathcal{A}^{\Sigma_2} \in \mathbf{A}_2\}$. □

3.1 Equality

Definition 14. Let Σ be a signature. The MODEL CLASS OF EQUALITY over Σ is the model class $M_{\approx}^{\Sigma} = (\Sigma, \mathbf{A})$, where $\mathbf{A}$ is the class of all Σ-structures over $\emptyset$. □

For any signature Σ, the satisfiability problem of $M_{\approx}^{\Sigma}$ is decidable [1].

Proposition 15. *Let φ be a satisfiable Σ-formula, where $\Sigma^{\mathrm{S}} = \{\sigma_1, \ldots, \sigma_n\}$. Morevoer, let $\kappa_1, \ldots, \kappa_n$ be infinite cardinal numbers. Then there exists a Σ-interpretation $\mathcal{A}$ such that $\mathcal{A} \models \varphi$ and $|A_{\sigma_i}| = \kappa_i$, for all $i = 1, \ldots, n$.* □

PROOF. Immediate consequence of the politeness [7] of the model class $M_{\approx}^{\Sigma}$ of equality. ■

3.2 Integers

The model class of integers M_{int} has a signature Σ_{int} containing a sort int for integers, plus the following symbols:

- the constant symbols 0 and 1, of sort int;
- the function symbols $+$, $-$, max, and min, of sort $\mathsf{int} \times \mathsf{int} \to \mathsf{int}$;[7]
- the predicate symbol $<$, of sort $\mathsf{int} \times \mathsf{int}$.

Definition 16. The STANDARD int-STRUCTURE $\mathcal{A}$ is the unique Σ_{int}-structure over $\emptyset$ satisfying the following conditions:

- $A_{\mathsf{int}} = \mathbb{Z}$;
- the symbols 0, 1, $+$, $-$, max, min, and $<$ are interpreted according to their standard interpretation over the integers.

The MODEL CLASS OF INTEGERS is the pair $M_{\mathsf{int}} = (\Sigma_{\mathsf{int}}, \mathbf{A})$, where $\mathbf{A}$ is the class of all Σ_{int}-structures that are isomorphic to the standard int-structure. $\square$

The satisfiability problem of M_{int} is decidable [6].

3.3 Lists

Let A be a nonempty set. A *list* x over A is a sequence $\langle a_1, \ldots, a_n \rangle$, where $n \geq 0$ and $\{a_1, \ldots, a_n\} \subseteq A$. We denote with A^* the set of lists over A.

In this section we define two model classes modeling lists:

- a model class M_{cons} of linear, flat, acyclic lists built using the constructors nil and cons;
- a model class M_{list} which extends M_{cons} with the selectors car and cdr.

The model class M_{cons} has a signature Σ_{cons} containing a sort elem for elements and a sort list for lists of elements, plus the following symbols:

- the constant symbol nil, of sort list;
- the function symbol cons, of arity $\mathsf{elem} \times \mathsf{list} \to \mathsf{list}$.

The model class M_{list} has a signature Σ_{list} that extends Σ_{cons} with the function symbols:

- car, of arity $\mathsf{list} \to \mathsf{elem}$;
- cdr, of arity $\mathsf{list} \to \mathsf{list}$.

Definition 17. A STANDARD cons-STRUCTURE $\mathcal{A}$ is a Σ_{cons}-structure over $\emptyset$ satisfying the following conditions:

- $A_{\mathsf{list}} = (A_{\mathsf{elem}})^*$;

[7] Although the symbols max and min can be expressed using $<$ and boolean connectives, we include them in order to conveniently define later the model class of multisets.

– $\mathsf{nil}^{\mathcal{A}} = \langle\rangle$;
– $\mathsf{cons}^{\mathcal{A}}(e, \langle e_1, \ldots, e_n\rangle) = \langle e, e_1, \ldots, e_n\rangle$, for each $n \geq 0$ and $e, e_1, \ldots, e_n \in A_{\mathsf{elem}}$.

The MODEL CLASS OF CONSTRUCTORS is the pair $M_{\mathsf{cons}} = (\Sigma_{\mathsf{cons}}, \mathbf{A})$, where $\mathbf{A}$ is the class of all Σ_{cons}-structures that are isomorphic to standard cons-structures. $\square$

Definition 18. A STANDARD list-STRUCTURE $\mathcal{A}$ is a Σ_{list}-structure over $\emptyset$ satisfying the following conditions:

– $\mathcal{A}^{\Sigma_{\mathsf{cons}}}$ is a standard cons-structure;
– $\mathsf{car}^{\mathcal{A}}(\langle e_1, \ldots, e_n\rangle) = e_1$, for each $n > 0$ and $e_1, \ldots, e_n \in A_{\mathsf{elem}}$;
– $\mathsf{cdr}^{\mathcal{A}}(\langle e_1, \ldots, e_n\rangle) = \langle e_2, \ldots, e_n\rangle$, for each $n > 0$ and $e_1, \ldots, e_n \in A_{\mathsf{elem}}$.

The MODEL CLASS OF LISTS is the pair $M_{\mathsf{list}} = (\Sigma_{\mathsf{list}}, \mathbf{A})$, where $\mathbf{A}$ is the class of all Σ_{list}-structures that are isomorphic to standard list-structures. $\square$

Note that for any list-structure $\mathcal{A}$, Definition 18 leaves underspecified the values of $\mathsf{car}^{\mathcal{A}}(\langle\rangle)$ and $\mathsf{cdr}^{\mathcal{A}}(\langle\rangle)$.

The satisfiability problems of both M_{cons} and M_{list} are decidable [2].

3.4 Arrays

The model class of arrays M_{array} has a signature Σ_{array} containing a sort elem for elements, a sort index for indices, and a sort array for arrays, plus the following two function symbols:

– read, of sort $\mathsf{array} \times \mathsf{index} \to \mathsf{elem}$;
– write, of sort $\mathsf{array} \times \mathsf{index} \times \mathsf{elem} \to \mathsf{array}$.

Notation. Given $a : I \to E$, $i \in I$ and $e \in E$, we define $a_{i \mapsto e} : I \to E$ as follows: $a_{i \mapsto e}(i) = e$ and $a_{i \mapsto e}(j) = a(j)$, for $j \neq i$.

Definition 19. A STANDARD array-STRUCTURE $\mathcal{A}$ is a Σ_{array}-structure satisfying the following conditions:

– $A_{\mathsf{array}} = (A_{\mathsf{elem}})^{A_{\mathsf{index}}}$;
– $\mathsf{read}^{\mathcal{A}}(a, i) = a(i)$, for each $a \in A_{\mathsf{array}}$ and $i \in A_{\mathsf{index}}$;
– $\mathsf{write}^{\mathcal{A}}(a, i, e) = a_{i \mapsto e}$, for each $a \in A_{\mathsf{array}}$, $i \in A_{\mathsf{index}}$, and $e \in A_{\mathsf{elem}}$.

The MODEL CLASS OF ARRAYS is the pair $M_{\mathsf{array}} = (\Sigma_{\mathsf{array}}, \mathbf{A})$, where $\mathbf{A}$ is the class of all Σ_{array}-structures that are isomorphic to standard array-structures. $\square$

The satisfiability problem of M_{array} is decidable [9].

3.5 Sets

The model class of sets M_{set} has a signature Σ_{set} containing a sort elem for elements and a sort set for sets of elements, plus the following symbols:

- the constant symbol $\emptyset$, of sort set;
- the function symbols:
  - $\{\cdot\}$, of sort $\mathsf{elem} \to \mathsf{set}$;
  - $\cup$, $\cap$, and $\setminus$, of sort $\mathsf{set} \times \mathsf{set} \to \mathsf{set}$;
- the predicate symbol $\in$, of sort $\mathsf{elem} \times \mathsf{set}$.

Definition 20. A STANDARD set-STRUCTURE $\mathcal{A}$ is a Σ_{set}-structure over $\emptyset$ satisfying the following conditions:

- $A_{\mathsf{set}} = \mathcal{P}(A_{\mathsf{elem}})$;
- the symbols $\emptyset$, $\{\cdot\}$, $\cup$, $\cap$, $\setminus$, and $\in$ are interpreted according to their interpretation structure over sets.

The MODEL CLASS OF SETS is the pair $M_{\mathsf{set}} = (\Sigma_{\mathsf{set}}, \mathbf{A})$, where $\mathbf{A}$ is the class of all Σ_{set}-structures that are isomorphic to standard set-structures. $\square$

The satisfiability problem of M_{set} is decidable [12].

3.6 Multisets

Multisets—also known as bags—are collections that may contain duplicate elements. Formally, a multiset x is a function $x : A \to \mathbb{N}$, for some set A.

We use the symbol $[\![\,]\!]$ to denote the empty multiset. When $n \geq 0$, we write $[\![e]\!]^{(n)}$ to denote the multiset containing exactly n occurrences of e and nothing else. When $n < 0$, we let $[\![e]\!]^{n} = [\![\,]\!]$.

Let x, y be two multisets. Then:

- their *union* $x \sqcup y$ is the multiset z such that, for each element e, the equality $z(e) = \max(x(e), y(e))$ holds;
- their *sum* $x \uplus y$ is the multiset z such that, for each element e, the equality $z(e) = x(e) + y(e)$ holds;
- their *intersection* $x \sqcap y$ is the multiset z such that, for each element e, the equality $z(e) = \min(x(e), y(e))$ holds.

The model class of multisets M_{bag} has a signature Σ_{bag} extending Σ_{int} with a sort elem for elements, and a sort bag for multisets, plus the following symbols:

- the constant symbol $[\![\,]\!]$, of sort bag;
- the function symbols:
  - $[\![\cdot]\!]^{(\cdot)}$, of sort $\mathsf{elem} \times \mathsf{int} \to \mathsf{bag}$;
  - $\sqcup$, $\uplus$, and $\sqcap$, of sort $\mathsf{bag} \times \mathsf{bag} \to \mathsf{bag}$;
  - count, of sort $\mathsf{elem} \times \mathsf{bag} \to \mathsf{int}$.

Definition 21. A STANDARD bag-STRUCTURE $\mathcal{A}$ is a Σ_{bag}-structure over $\emptyset$ satisfying the following conditions:

– $\mathcal{A}^{\Sigma_{\mathsf{int}}}$ is the standard int-structure;
– $A_{\mathsf{bag}} = \mathbb{N}^{A_{\mathsf{elem}}}$;
– the symbol $[\![\,]\!]$, $[\![\cdot]\!]^{(\cdot)}$, $\sqcup$, $\uplus$, and $\sqcap$ are interpreted according to their standard interpretation over multisets;
– $\mathsf{count}^{\mathcal{A}}(e, x) = x(e)$, for each $e \in A_{\mathsf{elem}}$ and $x \in A_{\mathsf{bag}}$.

The MODEL CLASS OF MULTISETS is the pair $M_{\mathsf{bag}} = \langle \Sigma_{\mathsf{bag}}, \mathbf{A} \rangle$, where $\mathbf{A}$ is the class of all Σ_{bag}-structures that are isomorphic to standard bag-structures. $\square$

The satisfiability problem of M_{bag} can be decided using a reduction to the satisfiability problem of M_{int} [11].

4 Ringeissen-Tinelli-Harandi Theorem

Our method for combining reduction functions relies on the Ringeissen-Tinelli-Harandi Theorem, a fundamental model-theoretical combination result independenlty discovered by Ringeissen [8] and Tinelli and Harandi [10].

Theorem 22 (Ringeissen-Tinelli-Harandi). *For $i = 1, 2$, let M_i be a Σ_i-model class, let φ_i be a Σ_i-formula, and let $X_i = free(\varphi_i)$. Also, let $\Sigma_0 = \Sigma_1 \cap \Sigma_2$ and $X_0 = X_1 \cap X_2$. Assume that there exist a Σ_1-structure $\mathcal{A}$ over X_1, and a Σ_2-structure $\mathcal{B}$ over X_2 such that:*

$$\mathcal{A} \models_{M_1} \varphi_1 \,, \qquad \mathcal{B} \models_{M_2} \varphi_2 \,, \qquad \mathcal{A}^{\Sigma_0, X_0} \cong \mathcal{B}^{\Sigma_0, X_0} \,.$$

Then there exists a $(\Sigma_1 \cup \Sigma_2)$-structure $\mathcal{F}$ over $X_1 \cup X_2$ such that:

$$\mathcal{F} \models_{M_1 \oplus M_2} \varphi_1 \wedge \varphi_2 \,, \qquad \mathcal{F}^{\Sigma_1, X_1} \cong \mathcal{A} \,, \qquad \mathcal{F}^{\Sigma_2, X_2} \cong \mathcal{B} \,.$$

$\square$

PROOF. Let h be an isomorphism of $\mathcal{A}^{\Sigma_0, X_0}$ into $\mathcal{B}^{\Sigma_0, X_0}$. By Proposition 12, we can assume without loss of generality that $\mathcal{A}^{\Sigma_0, X_0} = \mathcal{B}^{\Sigma_0, X_0}$. In particular, this implies that $A_\sigma = B_\sigma$, for all $\sigma \in \Sigma_0$.

We define a $(\Sigma_1 \cup \Sigma_2)$-structure $\mathcal{F}$ over $X_1 \cup X_2$ by letting:

$$F_\sigma = \begin{cases} A_\sigma \,, & \text{if } \sigma \in \Sigma_1^{\mathrm{S}} \,, \\ B_\sigma \,, & \text{if } \sigma \in \Sigma_2^{\mathrm{S}} \setminus \Sigma_1^{\mathrm{S}} \,, \end{cases}$$

and:

– for function symbols:

$$f^{\mathcal{F}} = \begin{cases} f^{\mathcal{A}}\,, & \text{if } f \in \Sigma_1^{\mathrm{F}}\,, \\ f^{\mathcal{B}}\,, & \text{if } f \in \Sigma_2^{\mathrm{F}} \setminus \Sigma_1^{\mathrm{F}}\,, \end{cases}$$

– for predicate symbols:

$$p^{\mathcal{F}} = \begin{cases} p^{\mathcal{A}}\,, & \text{if } f \in \Sigma_1^{\mathrm{P}}\,, \\ p^{\mathcal{B}}\,, & \text{if } p \in \Sigma_2^{\mathrm{P}} \setminus \Sigma_1^{\mathrm{P}}\,, \end{cases}$$

– for free symbols:

$$u^{\mathcal{F}} = \begin{cases} u^{\mathcal{A}}\,, & \text{if } u \in X_1\,, \\ u^{\mathcal{B}}\,, & \text{if } u \in X_2 \setminus X_1\,. \end{cases}$$

By construction, $\mathcal{F}^{\Sigma_1,X_1} \cong \mathcal{A}$ and $\mathcal{F}^{\Sigma_2,X_2} \cong \mathcal{B}$. Thus, by Proposition 12, $\mathcal{F} \models_{M_1 \oplus M_2} \varphi_1 \wedge \varphi_2$. ■

5 Normalization functions

We use normalization functions in order to construct purification functions (cf. Section 6 and Figure 2) and reduction functions (cf. Section 7 and Figure 3).

Definition 23. A formula is NORMALIZED if it is of the form

$$\begin{array}{lll} u\,, & u \leftrightarrow \neg v\,, & u \leftrightarrow (v \wedge w)\,, \\ u \leftrightarrow x \approx y\,, & u \leftrightarrow p(y_1, \ldots y_n)\,, & x \approx f(y_1, \ldots y_n)\,, \end{array}$$

where u, v, w are free propositional symbols, $x, y, y_1, \ldots, y_n$ are free constant symbols, f is a function symbol, and p is a predicate symbol. □

Intuitively, a normalization function translates a Σ-formula φ into a conjunction Δ of normalized Σ-formulae such that φ is satisfiable if and only if so is Δ.

Definition 24. A NORMALIZATION FUNCTION ν is a computable function such that:

– ν takes as input a Σ-formula φ;
– ν returns a conjunction Δ of normalized Σ-formulae such that $\mathit{free}(\varphi) \subseteq \mathit{free}(\Delta)$.

Moreover, if $\nu(\varphi) = \Delta$, $X = \mathit{free}(\varphi)$, $Y = \mathit{free}(\Delta)$, and M is a Σ-model class then:

(a) If $\mathcal{A} \models_M \varphi$, for some Σ-structure $\mathcal{A}$ over X, then there exists a Σ-structure $\mathcal{B}$ over Y such that

$$\mathcal{B} \models_M \Delta\,,$$
$$\mathcal{B}^{\Sigma,X} \cong \mathcal{A}\,.$$

$$\frac{\varphi[\neg u] \;\triangleright\; \Gamma}{\varphi[w] \;\triangleright\; \Gamma \wedge (w \leftrightarrow \neg u)} \qquad \frac{\varphi[u \wedge v] \;\triangleright\; \Gamma}{\varphi[w] \;\triangleright\; \Gamma \wedge (w \leftrightarrow (u \wedge v))}$$

$$\frac{\varphi[u \vee v] \;\triangleright\; \Gamma}{\varphi[\neg(\neg u \wedge \neg v)] \;\triangleright\; \Gamma} \qquad \frac{\varphi[u \rightarrow v] \;\triangleright\; \Gamma}{\varphi[\neg u \vee v] \;\triangleright\; \Gamma} \qquad \frac{\varphi[u \leftrightarrow v] \;\triangleright\; \Gamma}{\varphi[u \rightarrow v \wedge v \rightarrow u] \;\triangleright\; \Gamma}$$

$$\frac{\varphi[\mathit{if}\ u\ \mathit{then}\ v_1\ \mathit{else}\ v_2] \;\triangleright\; \Gamma}{\varphi[(u \rightarrow v_1) \wedge (\neg u \rightarrow v_2)] \;\triangleright\; \Gamma} \qquad \frac{\varphi[p(x_1, \ldots, x_n)] \;\triangleright\; \Gamma}{\varphi[w] \;\triangleright\; \Gamma \wedge (w \leftrightarrow p(x_1, \ldots, x_n))}$$

$$\frac{\varphi[c] \;\triangleright\; \Gamma}{\varphi[w] \;\triangleright\; \Gamma \wedge w \approx c} \qquad \frac{\varphi[f(x_1, \ldots, x_n)] \;\triangleright\; \Gamma}{\varphi[w] \;\triangleright\; \Gamma \wedge w \approx f(x_1, \ldots, x_n)}$$

Note:

- u, v, v_1, v_2 are free propositional symbols.
- $x_1, \ldots, x_n$ are free constant symbols.
- c is a constant symbol in Σ^{F}.
- f is a function symbol in Σ^{F}.
- p is a predicate symbol in Σ^{P}.
- w is a *fresh* free symbol.

Figure 1: Normalization rules.

(b) If $\mathcal{B} \models_T \Delta$, for some Σ-structure $\mathcal{B}$ over Y then

$$\mathcal{B}^{\Sigma,X} \models_M \varphi\,. \qquad \square$$

Normalization functions can be computed in linear time as follows. Given a Σ-formula φ, construct the pair $\varphi \triangleright \Gamma$, where Γ is the empty conjunction. Then, exhaustively apply the rules in Figure 1. Upon termination, we obtain a pair of the form $u \triangleright \Delta$, where u is a free propositional symbol, and Δ is a conjunction of normalized Σ-formulae. Clearly, $\nu(\varphi) = u \wedge \Delta$ is a normalization function.

Example 25. Consider the model classes M_{set} and M_{bag}, and let φ be the following $(\Sigma_{\mathsf{set}} \cup \Sigma_{\mathsf{bag}})$-formula:

$$\varphi : \qquad \{e_1\} \approx \{e_2\} \;\wedge\; [\![e_1]\!]^{(1)} \not\approx [\![e_2]\!]^{(1)}\,.$$

Then $\nu(\varphi)$ is equal to the following conjunction Γ of normalized $(\Sigma_{\mathsf{set}} \cup \Sigma_{\mathsf{bag}})$-formulae:

$$\Gamma = \left\{ \begin{array}{l} x_1 \approx \{e_1\}\,, \\ x_2 \approx \{e_2\}\,, \\ y_1 \approx [\![e_1]\!]^{(1)}\,, \\ y_2 \approx [\![e_2]\!]^{(1)}\,, \\ u_1 \leftrightarrow x_1 \approx x_2\,, \\ u_2 \leftrightarrow y_1 \approx y_2\,, \\ u_3 \leftrightarrow \neg u_2\,, \\ u_4 \leftrightarrow (u_1 \wedge u_3)\,, \\ u_4 \end{array} \right\}. \qquad \square$$

6 Purification functions

We use purification functions in order to combine reduction functions (cf. Sections 7 and 12, and Figure 4).

Let φ be a $(\Sigma_1 \cup \Sigma_2)$-formula. Intuitively, a purification function translates φ into a formula of the form $\varphi_1 \wedge \varphi_2$ such that:

- φ_i is a Σ_i-formula, for $i = 1, 2$;
- φ is satisfiable if and only if so is $\varphi_1 \wedge \varphi_2$.

Definition 26. Let Σ_1 and Σ_2 be signatures. A (Σ_1, Σ_2)-PURIFICATION FUNCTION π is a computable function such that

- π takes as input a $(\Sigma_1 \cup \Sigma_2)$-formula φ;
- π returns a $(\Sigma_1 \cup \Sigma_2)$-formula $\varphi_1 \wedge \varphi_2$ such that φ_i is a Σ_i-formula, for $i = 1, 2$, and $\mathit{free}(\varphi) \subseteq \mathit{free}(\varphi_1 \wedge \varphi_2)$.

Moreover, if $\pi(\varphi) = \varphi_1 \wedge \varphi_2$, $X = \mathit{free}(\varphi)$, $Y = \mathit{free}(\varphi_1 \wedge \varphi_2)$, and M is a $(\Sigma_1 \cup \Sigma_2)$-model class, then:

(a) If $\mathcal{A} \models_M \varphi$, for some $(\Sigma_1 \cup \Sigma_2)$-structure over X, then there exists a $(\Sigma_1 \cup \Sigma_2)$-structure $\mathcal{B}$ over Y such that

$$\mathcal{B} \models_M \varphi_1 \wedge \varphi_2\,,$$
$$\mathcal{B}^{\Sigma_1 \cup \Sigma_2, X} \cong \mathcal{A}\,.$$

(b) If $\mathcal{B} \models_M \varphi_1 \wedge \varphi_2$, for some $(\Sigma_1 \cup \Sigma_2)$-structure $\mathcal{B}$ over Y then

$$\mathcal{B}^{\Sigma_1 \cup \Sigma_2, X} \models_M \varphi\,. \qquad \square$$

Purification functions can be computed in linear time as follows. Given a $(\Sigma_1 \cup \Sigma_2)$-formula φ, compute $\Gamma = \nu(\varphi)$, where ν is a normalization function. Clearly, the formulae in Γ can be partitioned into two (possibly nondisjoint) sets Γ_1 and Γ_2 where, for $i = 1, 2$, Γ_i is the conjunction of all Σ_i-formulae occurring in Γ. Then, $\pi(\varphi) = \Gamma_1 \wedge \Gamma_2$ is a (Σ_1, Σ_2)-purification function. This process is depicted in Figure 2.

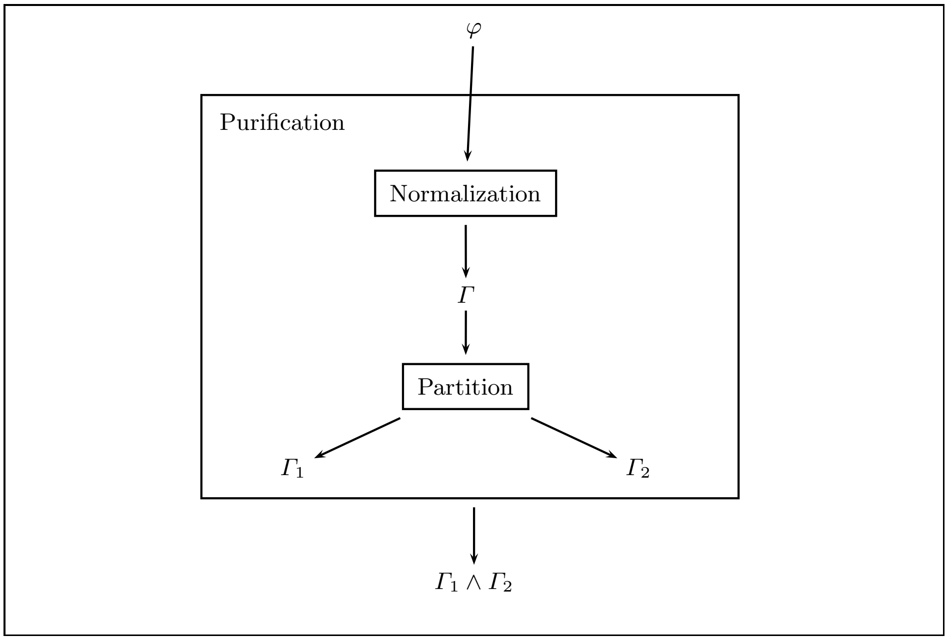

Figure 2: Computing purification functions.

Example 27. Consider the model classes M_{set} and M_{bag}, and let φ be the following $(\Sigma_{\mathsf{set}} \cup \Sigma_{\mathsf{bag}})$-formula:

$$\varphi : \qquad \{e_1\} \approx \{e_2\} \;\wedge\; [\![e_1]\!]^{(1)} \not\approx [\![e_2]\!]^{(1)}$$

Then $\pi(\varphi)$ is equal to $\Gamma_{\mathsf{set}} \wedge \Gamma_{\mathsf{bag}}$, where

$$\Gamma_{\mathsf{set}} = \left\{ \begin{array}{l} x_1 \approx \{e_1\}\,, \\ x_2 \approx \{e_2\}\,, \\ u_1 \leftrightarrow x_1 \approx x_2\,, \\ u_3 \leftrightarrow \neg u_2\,, \\ u_4 \leftrightarrow (u_1 \wedge u_3)\,, \\ u_4 \end{array} \right\}, \qquad \Gamma_{\mathsf{bag}} = \left\{ \begin{array}{l} y_1 \approx [\![e_1]\!]^{(1)}\,, \\ y_2 \approx [\![e_2]\!]^{(1)}\,, \\ u_2 \leftrightarrow y_1 \approx y_2\,, \\ u_3 \leftrightarrow \neg u_2\,, \\ u_4 \leftrightarrow (u_1 \wedge u_3)\,, \\ u_4 \end{array} \right\}. \qquad \square$$

7 Reduction functions

Let M be a Σ-model class, and let N be an Ω-model class such that $\Omega \subseteq \Sigma$ and $\Omega^{\mathrm{S}} = \Sigma^{\mathrm{S}}$. Intuitively, a reduction function from M to N translates a Σ-formula φ into an Ω-formula ψ such that φ is M-satisfiable if and only if ψ is N-satisfiable.

Definition 28. Let M be a Σ-model class, and let N be an Ω-model class such that $\Omega \subseteq \Sigma$ and $\Omega^{\mathrm{S}} = \Sigma^{\mathrm{S}}$. A REDUCTION FUNCTION from M to N is a computable binary function ρ such that:

– ρ takes as input a Σ-formula φ and a set $X_0 \subseteq \mathit{free}(\varphi)$;
– ρ outputs an Ω-formula ψ such that $\mathit{free}(\varphi) \subseteq \mathit{free}(\psi)$.

Moreover, if $\psi = \rho(\varphi, X_0)$, $X = \mathit{free}(\varphi)$, and $Y = \mathit{free}(\psi)$, then:

(a) If $\mathcal{A} \models_M \varphi$, for some Σ-structure $\mathcal{A}$ over X, then there exists an Ω-structure $\mathcal{B}$ over Y such that

$$\mathcal{B} \models_N \psi\,,$$
$$\mathcal{B}^{\Omega,X} \cong \mathcal{A}^{\Omega,X}\,.$$

(b) If $\mathcal{B} \models_N \psi$, for some Ω-structure $\mathcal{B}$ over Y, then there exists a Σ-structure $\mathcal{A}$ over X such that:

$$\mathcal{A} \models_M \varphi\,,$$
$$\mathcal{A}^{\Sigma^{\mathrm{S}},X_0} \cong \mathcal{B}^{\Sigma^{\mathrm{S}},X_0}\,.$$
□

Note that a reduction function takes in input *two* arguments. The second argument is needed in order to combine reduction functions (cf. Section 12, Figure 4, and Example 38).

Definition 28 clearly implies the following result.

Proposition 29. *Let M be a Σ-model class, and let N be an Ω-model class such that $\Omega \subseteq \Sigma$ and $\Omega^{\mathrm{S}} = \Sigma^{\mathrm{S}}$. Also, let ρ be a reduction function from M to N. Finally, let φ be a Σ-formula, and let $X_0 \subseteq \mathit{free}(\varphi)$. Then φ is M-satisfiable if and only if $\rho(\varphi, X_0)$ is N-satisfiable.* □

Definition 30. Let M be a Σ-model class, and let N be an Ω-model class such that $\Omega \subseteq \Sigma$ and $\Omega^{\mathrm{S}} = \Sigma^{\mathrm{S}}$. We say that M EFFECTIVELY REDUCES to N if there exists a reduction function from M to N. □

Proposition 31. *Let M be a Σ-model class, and let N be an Ω-model class such that $\Omega \subseteq \Sigma$ and $\Omega^{\mathrm{S}} = \Sigma^{\mathrm{S}}$. Assume that:*

– *M effectively reduces to N;*
– *the satisfiability problem of N is decidable.*

Then the satisfiability problem of M is decidable. □

PROOF. Just note that if φ is a Σ-formula and ρ is a reduction function from M to N, then φ is M-satisfiable if and only if $\rho(\varphi, \emptyset)$ is N-satisfiable. ■

Proposition 32. *Let M be a Σ-model class, and let N be an Ω-model class such that $\Omega \subseteq \Sigma$ and $\Omega^{\mathrm{S}} = \Sigma^{\mathrm{S}}$. Assume that there is a function r such that:*

– *r takes as input a conjunction Γ of normalized Σ-formulae and a set $X_0 \subseteq \mathit{free}(\Gamma)$;*
– *ρ outputs an Ω-formula ψ such that $\mathit{free}(\Gamma) \subseteq \mathit{free}(\psi)$.*

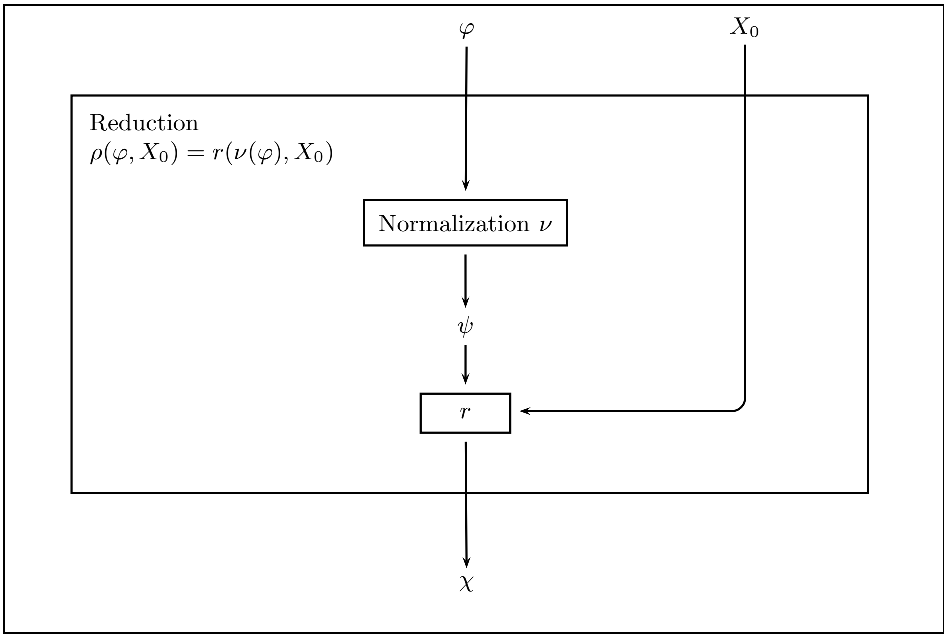

Figure 3: Computing reduction functions using normalization functions.

Moreover, assume that r satisfies conditions (a) and (b) of Definition 28. Then M effectively reduces to N. □

PROOF. Let φ be a Σ-formula, let $X_0 \subseteq \mathit{free}(\varphi)$, and let ν be a normalization function. Also, let

$$\begin{aligned} \nu(\varphi) &= \psi\,, \\ \chi &= r(\psi, X_0)\,, \\ X &= \mathit{free}(\varphi)\,, \\ Y &= \mathit{free}(\psi)\,, \\ Z &= \mathit{free}(\chi)\,, \end{aligned}$$

Note that $X_0 \subseteq X \subseteq Y \subseteq Z$. Consider the function ρ defined by letting

$$\rho(\varphi, X_0) = \chi\,,$$

and depicted in Figure 3.

We claim that ρ is a reduction function from M to N. We prove this claim by showing that ρ satisfies properties (a) and (b) of Definition 28.

(a) Assume that

$$\mathcal{A} \models_M \varphi\,,$$

for some Σ-structure $\mathcal{A}$ over X. By Definition 24, there exists a Σ-structure $\mathcal{B}$ over Y such that

$$\mathcal{B} \models_M \psi\,,$$
$$\mathcal{B}^{\Sigma,X} \cong \mathcal{A}\,.$$

By the properties of r, it follows that there exists an Ω-structure $\mathcal{C}$ over Z such that

$$\mathcal{C} \models_N \chi\,,$$
$$\mathcal{C}^{\Omega,Y} \cong \mathcal{B}^{\Omega,Y}\,,$$

which implies

$$\mathcal{C}^{\Omega,X} \cong \mathcal{A}^{\Omega,X}\,.$$

(b) Assume that

$$\mathcal{B} \models_N \chi\,,$$

for some Ω-structure $\mathcal{B}$ over Z. By the properties of r, there exists a Σ-structure $\mathcal{C}$ over Y such that

$$\mathcal{C} \models_M \psi\,,$$
$$\mathcal{C}^{\Sigma^{\mathrm{S}},X_0} \cong \mathcal{B}^{\Sigma^{\mathrm{S}},X_0}\,.$$

By Definition 24, it follows that

$$\mathcal{C}^{\Sigma,X} \models_M \varphi\,.$$

■

8 From lists to constructors

In this section we define a reduction function ρ_{list} that allows us to reduce the model class of lists to the model class of constructors.

8.1 The reduction function

Without loss of generality, let Γ be a conjunction of normalized Σ_{list}-formulae of the form:

$$\begin{array}{lll} u\,, & u \leftrightarrow \neg v\,, & u \leftrightarrow (v \wedge w)\,, \\ u \leftrightarrow e_1 \approx_{\mathsf{elem}} e_2\,, & u \leftrightarrow x \approx_{\mathsf{list}} y\,, & \\ x \approx \mathsf{nil}\,, & x \approx \mathsf{cons}(e, y)\,, & \\ e \approx \mathsf{car}(x)\,, & x \approx \mathsf{cdr}(y)\,, & \end{array}$$

where u, v, w are free propositional symbols, e, e_1, e_2 are free constant symbols of sort elem, and x, y are free constant symbols of sort list.

Let $X = \mathit{free}(\Gamma)$ and $X_0 \subseteq X$. We let $\rho_{\mathsf{list}}(\Gamma, X_0)$ be the conjunction obtained from Γ by means of the following process:

1. Replace each formula of the form $e \approx \mathsf{car}(x)$ in Γ with the formula

$$x \not\approx \mathsf{nil} \ \rightarrow \ x \approx \mathsf{cons}(e, y')\,,$$

where y' is a fresh free constant symbol of sort list.
2. Replace each formula of the form $x \approx \mathsf{cdr}(y)$ in Γ with the formula

$$y \not\approx \mathsf{nil} \ \rightarrow \ y \approx \mathsf{cons}(e', x)\,,$$

where e' is a fresh free constant symbol of sort elem.

8.2 The proof

Proposition 33. *M_{list} effectively reduces to M_{cons}.* □

PROOF. We want to show that ρ_{list} satisfies conditions (a) and (b) of Definition 28. To do so, let Γ be a conjunction of normalized Σ_{list}-formulae, let $X = \mathit{free}(\Gamma)$, and let $X_0 \subseteq X$. Finally, let $\Delta = \rho_{\mathsf{list}}(\Gamma, X_0)$ and $Y = \mathit{free}(\Delta)$.

Condition (a). Assume that $\Gamma^{\mathcal{A}} = \mathit{true}$, for some M_{list}-structure over X. Clearly, condition (a) follows by letting $\mathcal{B}$ be any M_{cons}-structure over Y constructed with the following process. First, we let

$$\mathcal{B}^{\Sigma_{\mathsf{cons}}, X} = \mathcal{A}^{\Sigma_{\mathsf{cons}}, X}$$

Then, if the literal

$$x \not\approx \mathsf{nil} \ \rightarrow \ x \approx \mathsf{cons}(e, y')$$

is in Δ, we let $(y')^{\mathcal{B}} = y_0$, provided that

$$[x \not\approx \mathsf{nil} \ \rightarrow \ x \approx \mathsf{cons}(e, y')]^{\mathcal{A}\{y'/y_0\}} = \mathit{true}\,.$$

Similarly, if the literal

$$y \not\approx \mathsf{nil} \ \rightarrow \ y \approx \mathsf{cons}(e', x)\,,$$

is in Δ, we let $(e')^{\mathcal{B}} = e_0$, provided that

$$[y \not\approx \mathsf{nil} \ \rightarrow \ y \approx \mathsf{cons}(e', x)]^{\mathcal{A}\{e'/e_0\}} = \mathit{true}\,.$$

Condition (b). Assume that $\Delta^{\mathcal{B}} = \mathit{true}$, for some M_{cons}-structure $\mathcal{B}$ over Y. Then condition (b) follows by letting $\mathcal{A}$ be any M_{list}-structure over X such that:

$$A_{\mathsf{elem}} = B_{\mathsf{elem}}\,,$$
$$A_{\mathsf{list}} = B_{\mathsf{list}}\,,$$

and

$$
\begin{aligned}
u^{\mathcal{A}} &= u^{\mathcal{B}}\,, && \text{for each } u \in X_{\mathsf{bool}}\,, \\
e^{\mathcal{A}} &= e^{\mathcal{B}}\,, && \text{for each } e \in X_{\mathsf{elem}}\,, \\
x^{\mathcal{A}} &= x^{\mathcal{B}}\,, && \text{for each } x \in X_{\mathsf{list}}\,.
\end{aligned}
$$

We show that all formulae in Γ are true in $\mathcal{A}$.

Formulae of the form u, $u \leftrightarrow \neg v$, and $u \leftrightarrow (v \wedge w)$. Immediate.

Formulae of the form $u \leftrightarrow e_1 \approx_{\mathsf{elem}} e_2$. Immediate.

Formulae of the form $u \leftrightarrow x \approx_{\mathsf{list}} y$. Just note that, by replacement 2 in Subsection 8.1, we have $x^{\mathcal{A}} = y^{\mathcal{A}}$ iff $x^{\mathcal{B}} = y^{\mathcal{B}}$.

Formulae of the form $x \approx \mathsf{nil}$ and $x \approx \mathsf{cons}(e, y)$. Immediate.

Formulae of the form $e \approx \mathsf{car}(x)$. By replacement 1 in Subsection 8.1.

Formulae of the form $x \approx \mathsf{cdr}(y)$. By replacement 2 in Subsection 8.1.

To conclude, we need to show that $\mathcal{A}^{\Sigma^{\mathrm{S}}_{\mathsf{cons}}, X_0} \cong \mathcal{B}^{\Sigma^{\mathrm{S}}_{\mathsf{cons}}, X_0}$. To see this, it suffices to note that:

– By construction, for every $x, y \in X_{\mathsf{bool}}$, we have $x^{\mathcal{A}} = y^{\mathcal{A}}$ iff $x^{\mathcal{B}} = y^{\mathcal{B}}$.
– By construction, for every $e_1, e_2 \in X_{\mathsf{elem}}$, we have $e_1^{\mathcal{A}} = e_2^{\mathcal{A}}$ iff $e_1^{\mathcal{B}} = e_2^{\mathcal{B}}$.
– By construction, for every $x, y \in X_{\mathsf{list}}$, we have $x^{\mathcal{A}} = y^{\mathcal{A}}$ iff $x^{\mathcal{B}} = y^{\mathcal{B}}$.
– By construction $|A_{\mathsf{elem}}| = |B_{\mathsf{elem}}|$ and $|A_{\mathsf{list}}| = |B_{\mathsf{list}}|$. ■

9 From arrays to equality

In this section we define a reduction function ρ_{array} that allows us to reduce the model class of arrays to the model class of equality.

9.1 The reduction function

Without loss of generality, let Γ be a conjunction of normalized Σ_{array}-formulae of the form

$$
\begin{array}{lll}
u\,, & u \leftrightarrow \neg v\,, & u \leftrightarrow (v \wedge w)\,, \\
u \leftrightarrow e_1 \approx_{\mathsf{elem}} e_2\,, & u \leftrightarrow i \approx_{\mathsf{index}} j\,, & u \leftrightarrow a \approx_{\mathsf{array}} b\,, \\
e \approx \mathsf{read}(a, i)\,, & a \approx \mathsf{write}(b, i, e)\,, &
\end{array}
$$

where u, v, w are free propositional symbols, e_1, e_2 are free constants symbols of sort elem, i, j are free constant symbols of sort index, and a, b are free constant symbols of sort array.

Let $X = \mathit{free}(\Gamma)$ and $X_0 \subseteq X$. We let $\rho_{\mathsf{array}}(\Gamma, X_0)$ be the conjunction obtained from Γ by means of the following process:

1. For each distinct $a, b \in X_{\mathsf{array}}$, generate a fresh constant symbol $w_{a,b}$ of sort index. Let W be the set of freshly generated free constant symbols.
2. For each formula of the form $u \leftrightarrow a \approx_{\mathsf{array}} b$ in Γ, add to Γ the following formula
$$a \not\approx b \ \rightarrow \ \mathsf{read}(a, w_{a,b}) \not\approx \mathsf{read}(b, w_{a,b})\,.$$
3. Replace each formula of the form $a \approx \mathsf{write}(b, i, e)$ in Γ with the formula
$$\bigwedge_{j \in X_{\mathsf{index}} \cup W} \mathit{if} \ \ i \approx j \ \ \mathit{then} \ \ \mathsf{read}(a, j) \approx e \ \ \mathit{else} \ \ \mathsf{read}(a, j) \approx \mathsf{read}(b, j)\,.$$
4. For each distinct $a, b \in X_{\mathsf{array}} \cap X_0$, add to Γ the following formula
$$a \not\approx b \ \rightarrow \ \mathsf{read}(a, w_{a,b}) \not\approx \mathsf{read}(b, w_{a,b})\,.$$

9.2 The proof

Proposition 34. *M_{array} effectively reduces to $M_{\approx}^{\Omega}$, where $\Omega^{\mathrm{S}} = \{\mathsf{elem}, \mathsf{index}, \mathsf{array}\}$, $\Omega^{\mathrm{F}} = \{\mathsf{read}\}$, and $\Omega^{\mathrm{P}} = \emptyset$.* □

Proof. We want to show that ρ_{array} satisfies conditions (a) and (b) of Definition 28. To do so, let Γ be a conjunction of normalized Σ_{array}-formulae, let $X = \mathit{free}(\Gamma)$, and let $X_0 \subseteq X$. Finally, let $\Delta = \rho_{\mathsf{array}}(\Gamma, X_0)$ and $Y = \mathit{free}(\psi)$.

Condition (a). Assume that $\Gamma^{\mathcal{A}} = \mathit{true}$, for some M_{array}-structure over X. Clearly, condition (a) follows by letting $\mathcal{B}$ be any Ω-structure over Y constructed with the following process. First, we let
$$\mathcal{B}^{\Omega,X} = \mathcal{A}^{\Omega,X}\,.$$

Then, for each distinct $a, b \in X_{\mathsf{array}}$, we consider two cases:

1. If $a^{\mathcal{A}} = b^{\mathcal{A}}$ then $w_{a,b}^{\mathcal{B}}$ is any arbitrary element of A_{index}.
2. If instead $a^{\mathcal{A}} \neq b^{\mathcal{A}}$ then there exists an $i_0 \in A_{\mathsf{index}}$ such that
$$[\mathsf{read}(a, w_{a,b}) \not\approx \mathsf{read}(b, w_{a,b})]^{\mathcal{A}\{w_{a,b}/i_0\}} = \mathit{true}\,,$$
and we let $w_{a,b}^{\mathcal{B}} = i_0$.

Condition (b). Assume that $\Delta^{\mathcal{B}} = \mathit{true}$, for some $M_{\approx}^{\Omega}$-structure $\mathcal{B}$ over Y. By proposition 15, without loss of generality assume that $|B_{\mathsf{elem}}| = |B_{\mathsf{index}}| = |\mathbb{N}|$ and $|B_{\mathsf{array}}| = |(B_{\mathsf{elem}})^{B_{\mathsf{index}}}| = |\mathbb{R}|$. Then condition (b) follows by letting $\mathcal{A}$ by the unique M_{array}-structure over X such that:
$$A_{\mathsf{elem}} = B_{\mathsf{elem}}\,,$$
$$A_{\mathsf{index}} = B_{\mathsf{index}}\,,$$

and

$$
\begin{aligned}
u^{\mathcal{A}} &= u^{\mathcal{B}}\,, && \text{for each } u \in X_{\mathsf{bool}}\,,\\
e^{\mathcal{A}} &= e^{\mathcal{B}}\,, && \text{for each } e \in X_{\mathsf{elem}}\,,\\
i^{\mathcal{A}} &= i^{\mathcal{B}}\,, && \text{for each } i \in X_{\mathsf{index}}\,.
\end{aligned}
$$

Moreover, for each $a \in X_{\mathsf{array}}$ and $i \in A_{\mathsf{index}}$, we let

$$
a^{\mathcal{A}}(i) = \begin{cases} \mathsf{read}^{\mathcal{B}}(a^{\mathcal{B}}, i)\,, & \text{if } i \in Y^{\mathcal{B}}\,,\\ e_0\,, & \text{otherwise}\,, \end{cases}
$$

where e_0 is an arbitrarily fixed element in A_{elem}. Intuitively, we don't care what $a^{\mathcal{A}}(i)$ is, when i is not represented by some free constant symbol in Y.

We show that all formulae in Γ are true in $\mathcal{A}$.

Formulae of the form u, $u \leftrightarrow \neg v$, and $u \leftrightarrow (v \wedge w)$. Immediate.

Formulae of the form $u \leftrightarrow e_1 \approx_{\mathsf{elem}} e_2$ and $u \leftrightarrow i \approx_{\mathsf{index}} j$. Immediate.

Formulae of the form $u \leftrightarrow a \approx_{\mathsf{array}} b$. Just note that, by replacement 2 in Subsection 9.1, we have $a^{\mathcal{A}} = b^{\mathcal{A}}$ iff $a^{\mathcal{B}} = b^{\mathcal{B}}$.

Formulae of the form $e \approx \mathsf{read}(a, i)$. Just note that, by construction $[\mathsf{read}(a, i)]^{\mathcal{A}} = [\mathsf{read}(a, i)]^{\mathcal{B}}$.

Formulae of the form $a \approx \mathsf{write}(b, i, e)$. By replacement 3 in Subsection 9.1.

To conclude, we need to show that $\mathcal{A}^{\Omega^{\mathrm{S}}, X_0} \cong \mathcal{B}^{\Omega^{\mathrm{S}}, X_0}$. To see this, it suffices to note that:

– By construction, for every $x, y \in X_{\mathsf{bool}}$, we have $x^{\mathcal{A}} = y^{\mathcal{A}}$ iff $x^{\mathcal{B}} = y^{\mathcal{B}}$.
– By construction, for every $e_1, e_2 \in X_{\mathsf{elem}}$, we have $e_1^{\mathcal{A}} = e_2^{\mathcal{A}}$ iff $e_1^{\mathcal{B}} = e_2^{\mathcal{B}}$.
– By construction, for every $i, j \in X_{\mathsf{index}}$, we have $i^{\mathcal{A}} = j^{\mathcal{A}}$ iff $i^{\mathcal{B}} = j^{\mathcal{B}}$.
– By replacement 4 in Subsection 9.1, for every $a, b \in X_{\mathsf{array}} \cap X_0$, we have $a^{\mathcal{A}} = b^{\mathcal{A}}$ iff $a^{\mathcal{B}} = b^{\mathcal{B}}$.
– By construction $|A_{\mathsf{elem}}| = |B_{\mathsf{elem}}| = |\mathbb{N}|$ and $|A_{\mathsf{index}}| = |B_{\mathsf{index}}| = |\mathbb{N}|$. We also have $|A_{\mathsf{array}}| = \left|(A_{\mathsf{elem}})^{A_{\mathsf{index}}}\right| = |\mathbb{R}| = |B_{\mathsf{array}}|$. ∎

10 From sets to equality

In this section we define a reduction function ρ_{set} that allows us to reduce the model class of sets to the model class of equality.

10.1 The reduction function

Without loss of generality, let Γ be a conjunction of normalized Σ_{set}-formulae of the form

$$
\begin{array}{lll}
u\,, & u \leftrightarrow \neg v\,, & u \leftrightarrow (v \wedge w)\,, \\
u \leftrightarrow e_1 \approx_{\mathsf{elem}} e_2\,, & u \leftrightarrow x \approx_{\mathsf{set}} y\,, & u \leftrightarrow e \in x\,, \\
x \approx \emptyset\,, & x \approx \{e\}\,, & \\
x \approx y \cup z\,, & x \approx y \cap z\,, & x \approx y \setminus z\,,
\end{array}
$$

where u, v, w are free propositional symbols, e, e_1, e_2 are free constant symbols of sort elem, and x, y, z are free constant symbols of sort set.

Let $X = \mathit{free}(\Gamma)$ and $X_0 \subseteq X$. We let $\rho_{\mathsf{set}}(\Gamma, X_0)$ be the conjunction obtained from Γ by means of the following process:

1. For each distinct $x, y \in X_{\mathsf{set}}$, generate a fresh free constant symbol $w_{x,y}$ of sort elem. Let W be the set of freshly generated free constant symbols.
2. For each formula of the form $u \leftrightarrow x \approx_{\mathsf{set}} y$ in Γ, add to Γ the following formula
$$x \not\approx y \;\rightarrow\; ((w_{x,y} \in x \;\wedge\; w_{x,y} \notin y) \;\vee\; (w_{x,y} \notin x \;\wedge\; w_{x,y} \in y))\,.$$
3. Replace each formula of the form $x \approx \emptyset$ in Γ with the formula
$$\bigwedge_{e \in X_{\mathsf{elem}} \cup W} e \notin x\,.$$
4. Replace each formula of the form $x \approx \{e_0\}$ in Γ with the formula
$$\bigwedge_{e \in X_{\mathsf{elem}} \cup W} [e \in x \;\leftrightarrow\; e = e_0]\,,$$
5. Replace each formula of the form $x \approx y \cup z$ in Γ with the formula
$$\bigwedge_{e \in X_{\mathsf{elem}} \cup W} [e \in x \;\leftrightarrow\; (e \in y \;\vee\; e \in z)]\,.$$
6. Replace each formula of the form $x \approx y \cap z$ in Γ with the formula
$$\bigwedge_{e \in X_{\mathsf{elem}} \cup W} [e \in x \;\leftrightarrow\; (e \in y \;\wedge\; e \in z)]\,.$$
7. Replace each formula of the form $x \approx y \setminus z$ in Γ with the formula
$$\bigwedge_{e \in X_{\mathsf{elem}} \cup W} [e \in x \;\leftrightarrow\; (e \in y \;\wedge\; e \notin z)]\,.$$
8. For each distinct $x, y \in X_{\mathsf{set}} \cap X_0$, add to Γ the following formula
$$x \not\approx y \;\rightarrow\; ((w_{x,y} \in x \;\wedge\; w_{x,y} \notin y) \;\vee\; (w_{x,y} \notin x \;\wedge\; w_{x,y} \in y))\,.$$

10.2 The proof

Proposition 35. *M_{set} effectively reduces to $M^{\Omega}_{\approx}$, where $\Omega^{\mathrm{S}} = \{\mathsf{elem}, \mathsf{set}\}$, $\Omega^{\mathrm{F}} = \emptyset$, and $\Omega^{\mathrm{P}} = \{\in\}$.* □

Proof. We want to show that ρ_{set} satisfies conditions (a) and (b) of Definition 28. To do so, let Γ be a conjunction of normalized Σ_{set}-formulae, let $X = \mathit{free}(\Gamma)$, and let $X_0 \subseteq X$. Finally, let $\Delta = \rho_{\mathsf{set}}(\Gamma, X_0)$ and $Y = \mathit{free}(\Delta)$.

Condition (a). Assume that $\Gamma^{\mathcal{A}} = \mathit{true}$, for some M_{set}-structure over X. Clearly, condition (a) follows by letting $\mathcal{B}$ be any Ω-structure over Y constructed with the following process. First, we let

$$\mathcal{B}^{\Omega,X} = \mathcal{A}^{\Omega,X}$$

Then, for each distinct $x, y \in X_{\mathsf{set}}$, we consider two cases:

1. If $x^{\mathcal{A}} = y^{\mathcal{A}}$ then $w^{\mathcal{B}}_{x,y}$ is any arbitrary element of A_{elem}.
2. If instead $x^{\mathcal{A}} \neq y^{\mathcal{A}}$ then there exists an $e_0 \in A_{\mathsf{elem}}$ such that

   $$[(w_{x,y} \in x \ \wedge \ w_{x,y} \notin y) \ \vee \ (w_{x,y} \notin x \ \wedge \ w_{x,y} \in y)]^{\mathcal{A}\{w_{x,y}/e_0\}} = \mathit{true}\,,$$

   and we let $w^{\mathcal{B}}_{x,y} = e_0$.

Condition (b). Assume that $\Delta^{\mathcal{B}} = \mathit{true}$, for some $M^{\Omega}_{\approx}$-structure $\mathcal{B}$ over Y. By proposition 15, without loss of generality assume that $|B_{\mathsf{elem}}| = |\mathbb{N}|$ and $|B_{\mathsf{set}}| = |\mathcal{P}(B_{\mathsf{elem}})| = |\mathbb{R}|$. Then condition (b) follows by letting $\mathcal{A}$ be the unique M_{set}-structure over X such that

$$A_{\mathsf{elem}} = B_{\mathsf{elem}}\,,$$

and

$$\begin{aligned}
u^{\mathcal{A}} &= u^{\mathcal{B}}\,, && \text{for each } u \in X_{\mathsf{bool}}\,,\\
e^{\mathcal{A}} &= e^{\mathcal{B}}\,, && \text{for each } e \in X_{\mathsf{elem}}\,,\\
x^{\mathcal{A}} &= \big\{e \in Y^{\mathcal{A}} \mid \big(e, x^{\mathcal{B}}\big) \in \big(\in^{\mathcal{B}}\big)\big\}\,, && \text{for each } x \in X_{\mathsf{set}}\,.
\end{aligned}$$

Intuitively, if e is not represented by some free constant symbol in Y then we let $e \notin a^{\mathcal{A}}$. If instead $e \in Y^{\mathcal{B}}$ then we let $e \in a^{\mathcal{A}}$ iff $\big(e, a^{\mathcal{B}}\big) \in \big(\in^{\mathcal{B}}\big)$.

We show that all formulae in Γ are true in $\mathcal{A}$.

Formulae of the form u, $u \leftrightarrow \neg v$, and $u \leftrightarrow (v \wedge w)$. Immediate.

Formulae of the form $u \leftrightarrow e_1 \approx_{\mathsf{elem}} e_2$. Immediate.

Formulae of the form $u \leftrightarrow x \approx_{\mathsf{set}} y$. Just note that, by replacement 2 in Subsection 10.1, we have $x^{\mathcal{A}} = y^{\mathcal{A}}$ iff $x^{\mathcal{B}} = y^{\mathcal{B}}$.

Formulae of the form $u \leftrightarrow e \in x$. Just note that, by construction, $[e \in x]^{\mathcal{A}} = [e \in x]^{\mathcal{B}}$.

Formulae of the form $x \approx \emptyset$. By replacement 3 in Subsection 10.1.

Formulae of the form $x \approx \{e\}$. By replacement 4 in Subsection 10.1.

Formulae of the form $x \approx y \cup z$. By replacement 5 in Subsection 10.1.

Formulae of the form $x \approx y \cap z$. By replacement 6 in Subsection 10.1.

Formulae of the form $x \approx y \setminus z$. By replacement 7 in Subsection 10.1.

To conclude, we need to show that $\mathcal{A}^{\Omega^{\mathrm{S}}, X_0} \cong \mathcal{B}^{\Omega^{\mathrm{S}}, X_0}$. To see this, it suffices to note that:

- By construction, for every $x, y \in X_{\mathsf{bool}}$, we have $x^{\mathcal{A}} = y^{\mathcal{A}}$ iff $x^{\mathcal{B}} = y^{\mathcal{B}}$.
- By construction, for every $e_1, e_2 \in X_{\mathsf{elem}}$, we have $e_1^{\mathcal{A}} = e_2^{\mathcal{A}}$ iff $e_1^{\mathcal{B}} = e_2^{\mathcal{B}}$.
- By replacement 8 in Subsection 10.1, for every $x, y \in X_{\mathsf{set}} \cap X_0$, we have $x^{\mathcal{A}} = y^{\mathcal{A}}$ iff $x^{\mathcal{B}} = y^{\mathcal{B}}$.
- By construction $|A_{\mathsf{elem}}| = |B_{\mathsf{elem}}| = |\mathbb{N}|$. We also have $|A_{\mathsf{set}}| = |\mathcal{P}(A_{\mathsf{elem}})| = |\mathbb{R}| = |B_{\mathsf{set}}|$. ∎

11 From multisets to integers

In this section we define a reduction function ρ_{bag} that allows us to reduce the model class of multisets to the model class of integers extended with uninterpreted function symbols.

11.1 The reduction function

Without loss of generality, let Γ be a conjunction of normalized Σ_{bag}-formulae containing, besides normalized Σ_{int}-formulae, also formulae of the form

$$
\begin{array}{lll}
u \leftrightarrow e_1 \approx_{\mathsf{elem}} e_2 \,, & u \leftrightarrow x \approx_{\mathsf{bag}} y \,, & \\
x = [\![\,]\!] \,, & x = [\![e]\!]^{(m)} \,, & \\
x = y \sqcup z \,, & x = y \uplus z \,, & x = y \sqcap z \,, \\
m = \mathsf{count}(e, x) \,, & &
\end{array}
$$

where u is a free propositional constant, m is a free propositional constant of sort int, e, e_1, e_2 are free propositional constants of sort elem, and x, y, z are free constant symbols of sort bag.

Let $X = \mathit{free}_{\sigma}(\Gamma)$ and $X_0 \subseteq \mathit{free}(\Gamma)$. We let $\rho_{\mathsf{bag}}(\Gamma, X_0)$ be the conjunction obtained from Γ by means of the following process:

1. For each distinct $x, y \in X_{\mathsf{bag}}$, generate a fresh free constant symbol $w_{x,y}$ of sort elem. Let W be the set of freshly generated free constant symbols.
2. For each formula of the form $u \leftrightarrow x \approx_{\mathsf{bag}} y$ in Γ, add to Γ the following formula

$$x \not\approx y \;\rightarrow\; \mathsf{count}(w_{x,y}, x) \not\approx \mathsf{count}(w_{x,y}, y)\,.$$

3. Replace each formula of the form $x \approx [\![\,]\!]$ in Γ with the formula

$$\bigwedge_{e \in X_{\mathsf{elem}} \cup W} \mathsf{count}(e, x) \approx 0\,.$$

4. Replace each formula of the form $x \approx [\![e_0]\!]^{(u)}$ in Γ with the formula

$$\bigwedge_{e \in V_{\mathsf{elem}} \cup W} [\mathit{if}\;\; e \approx e_0 \;\;\mathit{then}\;\; \mathsf{count}(e, x) \approx \mathsf{max}(0, u) \;\;\mathit{else}\;\; \mathsf{count}(e, x) \approx 0]\,.$$

5. Replace each formula of the form $x \approx y \sqcup z$ in Γ with the formula

$$\bigwedge_{e \in X_{\mathsf{elem}} \cup W} [\mathsf{count}(e, x) \approx \mathsf{max}(\mathsf{count}(e, y), \mathsf{count}(e, z))]\,.$$

6. Replace each formula of the form $x \approx y \uplus z$ in Γ with the formula

$$\bigwedge_{e \in X_{\mathsf{elem}} \cup W} [\mathsf{count}(e, x) \approx \mathsf{count}(e, y) + \mathsf{count}(e, z))]\,.$$

7. Replace each formula of the form $x \approx y \sqcap z$ in Γ with the formula

$$\bigwedge_{e \in X_{\mathsf{elem}} \cup W} [\mathsf{count}(e, x) \approx \mathsf{min}(\mathsf{count}(e, y), \mathsf{count}(e, z))]\,.$$

8. For each distinct $x, y \in X_{\mathsf{bag}} \cap X_0$, add to Γ the following formula

$$x \not\approx y \;\rightarrow\; \mathsf{count}(w_{x,y}, x) \not\approx \mathsf{count}(w_{x,y}, y)\,.$$

11.2 The proof

Proposition 36. *M_{bag} effectively reduces to $M_{\mathsf{int}} \oplus M_{\approx}^{\Omega}$, where $\Omega^{\mathrm{S}} = \{\mathsf{int}, \mathsf{elem}, \mathsf{bag}\}$, $\Omega^{\mathrm{F}} = \{\mathsf{count}\}$, and $\Omega^{\mathrm{P}} = \emptyset$.* $\square$

Proof. We want to show that ρ_{bag} satisfies conditions (a) and (b) of Definition 28. To do so, let Γ be a conjunction of normalized Σ_{bag}-formulae, let $X = \mathit{free}(\Gamma)$, and let $X_0 \subseteq X$. Finally, let $\Delta = \rho_{\mathsf{bag}}(\Gamma, X_0)$ and $Y = \mathit{free}(\Delta)$.

Condition (a). Assume that $\Gamma^{\mathcal{A}} = \mathit{true}$, for some M_{bag}-structure over X. Clearly, condition (a) follows by letting $\mathcal{B}$ be any $(M_{\mathsf{int}} \oplus M_{\approx}^{\Omega})$-structure over Y constructed with the following process. First, we let

$$\mathcal{B}^{\Sigma_{\mathsf{int}} \cup \Omega, X} = \mathcal{A}^{\Sigma_{\mathsf{int}} \cup \Omega, X}\,.$$

Then, for each distinct $x, y \in X_{\mathsf{bag}}$, we consider two cases:

1. If $x^{\mathcal{A}} = y^{\mathcal{A}}$ then $w_{x,y}^{\mathcal{B}}$ is any arbitrary element of A_{elem}.
2. If instead $x^{\mathcal{A}} \neq y^{\mathcal{A}}$ then there exists an $e_0 \in A_{\mathsf{elem}}$ such that

$$[\mathsf{count}(w_{x,y}, x) \not\approx \mathsf{count}(w_{x,y}, y)]^{\mathcal{A}\{w_{a,b}/e_0\}} = \mathit{true}\,.$$

   and we let $w_{x,y}^{\mathcal{B}} = e_0$.

Condition (b). Assume that $\Delta^{\mathcal{B}} = \mathit{true}$, for some $(M_{\mathsf{int}} \oplus M_{\approx}^{\Omega})$-structure $\mathcal{B}$ over Y. By Proposition 15, without loss of generality, assume that $|B_{\mathsf{int}}| = |B_{\mathsf{elem}}| = \mathbb{N}$ and $|B_{\mathsf{bag}}| = |\mathbb{N}^{B_{\mathsf{elem}}}| = |\mathbb{R}|$. Then condition (b) follows by letting $\mathcal{A}$ be the unique M_{bag}-structure over X such that

$$A_{\mathsf{elem}} = B_{\mathsf{elem}}\,,$$

and

$$\begin{aligned} u^{\mathcal{A}} &= u^{\mathcal{B}}\,, && \text{for each } u \in X_{\mathsf{bool}}\,, \\ v^{\mathcal{A}} &= v^{\mathcal{B}}\,, && \text{for each } v \in X_{\mathsf{int}}\,, \\ e^{\mathcal{A}} &= e^{\mathcal{B}}\,, && \text{for each } e \in X_{\mathsf{elem}}\,. \end{aligned}$$

Moreover, for each $a \in X_{\mathsf{bag}}$ and $e \in A_{\mathsf{elem}}$, we let

$$a^{\mathcal{A}}(e) = \begin{cases} \mathsf{count}^{\mathcal{B}}(e, a^{\mathcal{B}})\,, & \text{if } e \in Y^{\mathcal{B}}\,, \\ 0\,, & \text{otherwise}\,. \end{cases}$$

Intuitively, if e is not represented by some free constant symbol in Y then we let $a^{\mathcal{A}}(e) = 0$. If instead $e \in Y^{\mathcal{B}}$ then we let $a^{\mathcal{A}}(e) = a^{\mathcal{B}}(e)$.

We show that all formulae in Γ are true in $\mathcal{A}$.

Normalized Σ_{int}-formulae. Immediate.

Formulae of the form u, $u \leftrightarrow \neg v$, and $u \leftrightarrow (v \wedge w)$. Immediate.

Formulae of the form $u \leftrightarrow e_1 \approx_{\mathsf{elem}} e_2$. Immediate.

Formulae of the form $u \leftrightarrow x \approx_{\mathsf{bag}} y$. Just note that, by replacement 2 in Subsection 11.1, we have $x^{\mathcal{A}} = y^{\mathcal{A}}$ iff $x^{\mathcal{B}} = y^{\mathcal{B}}$.

Formulae of the form $x \approx [\![\,]\!]$. By replacement 3 in Subsection 11.1.

Formulae of the form $x \approx [\![e]\!]^{(m)}$. By replacement 4 in Subsection 11.1.

Formulae of the form $x \approx y \cup z$. By replacement 5 in Subsection 11.1.

Formulae of the form $x \approx y \uplus z$. By replacement 6 in Subsection 11.1.

Formulae of the form $x \approx y \cap z$. By replacement 7 in Subsection 11.1.

Formulae of the form $u \approx \mathsf{count}(e, x)$. Just note that, by construction, $[\mathsf{count}(e, x)]^{\mathcal{A}} = [\mathsf{count}(e, x)]^{\mathcal{B}}$.

To conclude, we need to show that $\mathcal{A}^{\Omega^{\mathrm{S}}, X_0} \cong \mathcal{B}^{\Omega^{\mathrm{S}}, X_0}$. To see this, it suffices to note that:

– By construction, for every $x, y \in X_{\mathsf{bool}}$, we have $x^{\mathcal{A}} = y^{\mathcal{A}}$ iff $x^{\mathcal{B}} = y^{\mathcal{B}}$.
– By construction, for every $e_1, e_2 \in X_{\mathsf{elem}}$, we have $e_1^{\mathcal{A}} = e_2^{\mathcal{A}}$ iff $e_1^{\mathcal{B}} = e_2^{\mathcal{B}}$.
– By construction, for every $m_1, m_2 \in X_{\mathsf{int}}$, we have $m_1^{\mathcal{A}} = m_2^{\mathcal{A}}$ iff $m_1^{\mathcal{B}} = m_2^{\mathcal{B}}$.
– By replacement 8 in Subsection 11.1, for every $x, y \in X_{\mathsf{bag}} \cap X_0$, we have $x^{\mathcal{A}} = y^{\mathcal{A}}$ iff $x^{\mathcal{B}} = y^{\mathcal{B}}$.
– By construction $|A_{\mathsf{int}}| = |B_{\mathsf{int}}| = |\mathbb{N}|$ and $|A_{\mathsf{elem}}| = |B_{\mathsf{elem}}| = |\mathbb{N}|$. We also have $|A_{\mathsf{bag}}| = |\mathbb{N}^{A_{\mathsf{elem}}}| = |\mathbb{R}| = |B_{\mathsf{bag}}|$. ■

12 Combining reduction functions

12.1 The combination method

Let us be given the following model classes:

$$M_1 = (\Sigma_1, \mathbf{A}_1), \qquad N_1 = (\Omega_1, \mathbf{B}_1),$$
$$M_2 = (\Sigma_2, \mathbf{A}_2), \qquad N_2 = (\Omega_2, \mathbf{B}_2).$$

Assume that:

– $\Sigma_1^{\mathrm{F}} \cap \Sigma_2^{\mathrm{F}} = \emptyset$;
– $\Sigma_1^{\mathrm{P}} \cap \Sigma_2^{\mathrm{P}} = \emptyset$;
– $\Omega_1 \subseteq \Sigma_1$ and $\Omega_1^{\mathrm{S}} = \Sigma_1^{\mathrm{S}}$;
– $\Omega_2 \subseteq \Sigma_2$ and $\Omega_2^{\mathrm{S}} = \Sigma_2^{\mathrm{S}}$;
– ρ_1 is a reduction function from M_1 to N_1;
– ρ_2 is a reduction function from M_2 to N_2.

Moreover, let π be a (Σ_1, Σ_2)-purification function. Then a reduction function ρ from $M_1 \oplus M_2$ to $N_1 \oplus N_2$ can be constructed by letting:

$$\rho(\varphi, X_0) = \rho_1(\varphi_1, X_1') \wedge \rho_2(\varphi_2, X_2'),$$

where

$$\begin{aligned} \pi(\varphi) &= \varphi_1 \wedge \varphi_2, \\ X_1 &= \mathit{free}(\varphi_1) \\ X_2 &= \mathit{free}(\varphi_2) \\ X_1' &= X_1 \cap (X_0 \cup X_2), \\ X_2' &= X_2 \cap (X_0 \cup X_1). \end{aligned}$$

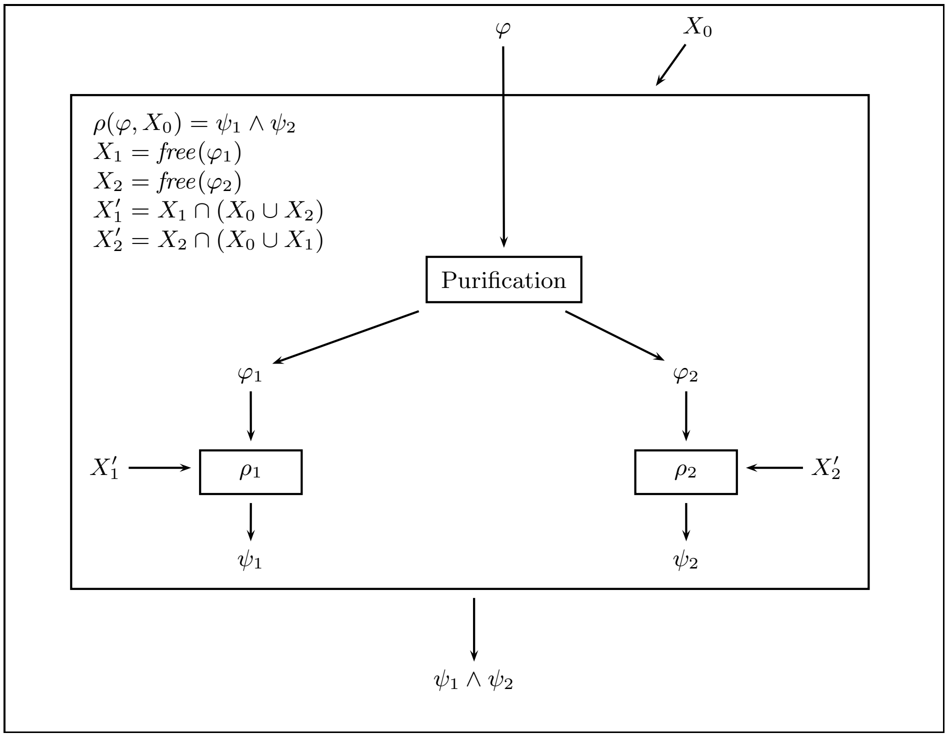

Figure 4: Combining reduction functions.

Note that $X_i' = (X_0 \cap X_i) \cup (X_1 \cap X_2)$. In other words, X_i' contains all the free constant symbols that are both in X_0 and X_i, as well as the "shared" free constant symbols in $X_1 \cap X_2$.

The process of combining reduction functions is depicted in Figure 4.

Example 37. Consider the model classes M_{set} and M_{bag}, and let φ be the following $(\Sigma_{\mathsf{set}} \cup \Sigma_{\mathsf{bag}})$-formula:

$$\varphi: \qquad \{e_1\} \approx \{e_2\} \ \wedge\ [\![e_1]\!]^{(1)} \not\approx [\![e_2]\!]^{(1)} .$$

We want to use the reduction approach in order to show that φ is $(M_{\mathsf{set}} \oplus M_{\mathsf{bag}})$-unsatisfiable.

Specifically, we combine the reduction functions ρ_{set} and ρ_{bag} in order to obtain a reduction function ρ from the model class $M_{\mathsf{set}} \oplus M_{\mathsf{bag}}$ to the model class $M_{\mathsf{int}} \oplus M_{\approx}^{\Omega}$, where $\Omega = (\{\mathsf{int}, \mathsf{elem}, \mathsf{set}, \mathsf{bag}\}, \{\mathsf{count}\}, \{\in\})$.

From Examples 25 and 27, we know that $\pi(\varphi) = \Gamma_{\mathsf{set}} \wedge \Gamma_{\mathsf{bag}}$, where

$$\Gamma_{\mathsf{set}} = \left\{ \begin{array}{l} x_1 \approx \{e_1\}\,, \\ x_2 \approx \{e_2\}\,, \\ u_1 \leftrightarrow x_1 \approx x_2\,, \\ u_3 \leftrightarrow \neg u_2\,, \\ u_4 \leftrightarrow (u_1 \wedge u_3)\,, \\ u_4 \end{array} \right\}, \qquad \Gamma_{\mathsf{bag}} = \left\{ \begin{array}{l} y_1 \approx [\![e_1]\!]^{(1)}\,, \\ y_2 \approx [\![e_2]\!]^{(1)}\,, \\ u_2 \leftrightarrow y_1 \approx y_2\,, \\ u_3 \leftrightarrow \neg u_2\,, \\ u_4 \leftrightarrow (u_1 \wedge u_3)\,, \\ u_4 \end{array} \right\}.$$

Therefore, we have

$$\rho(\varphi, \emptyset) = \rho_{\mathsf{set}}(\Gamma_{\mathsf{set}}, \{e_1, e_2\}) \wedge \rho_{\mathsf{bag}}(\Gamma_{\mathsf{bag}}, \{e_1, e_2\})\,.$$

In particular, we have

$$\rho_{\mathsf{set}}(\Gamma_{\mathsf{set}}, \{e_1, e_2\}) = \left\{ \begin{array}{l} x_1 \not\approx x_2 \;\rightarrow\; ((e_3 \in x \;\wedge\; e_3 \notin x) \;\vee\; ((e_3 \notin x) \wedge (e_3 \in y))\,, \\ e_1 \in x_1 \;\leftrightarrow\; e_1 \approx e_1\,, \\ e_2 \in x_1 \;\leftrightarrow\; e_2 \approx e_1\,, \\ e_3 \in x_1 \;\leftrightarrow\; e_3 \approx e_1\,, \\ e_1 \in x_2 \;\leftrightarrow\; e_1 \approx e_2\,, \\ e_2 \in x_2 \;\leftrightarrow\; e_2 \approx e_2\,, \\ e_3 \in x_2 \;\leftrightarrow\; e_3 \approx e_2\,, \\ u_1 \leftrightarrow x_1 \approx x_2\,, \\ u_3 \leftrightarrow \neg u_2\,, \\ u_4 \leftrightarrow (u_1 \wedge u_3)\,, \\ u_4 \end{array} \right\}$$

and

$$\rho_{\mathsf{bag}}(\Gamma_{\mathsf{bag}}, \{e_1, e_2\}) = \left\{ \begin{array}{l} y_1 \not\approx y_2 \;\rightarrow\; \mathsf{count}(e_4, y_1) \not\approx \mathsf{count}(e_4, y_2)\,, \\ \textit{if}\;\; e_1 \approx e_1 \;\textit{then}\; \mathsf{count}(e_1, y_1) \approx \mathsf{max}(0,1) \;\textit{else}\; \mathsf{count}(e_1, y_1) \approx 0\,, \\ \textit{if}\;\; e_2 \approx e_1 \;\textit{then}\; \mathsf{count}(e_2, y_1) \approx \mathsf{max}(0,1) \;\textit{else}\; \mathsf{count}(e_2, y_1) \approx 0\,, \\ \textit{if}\;\; e_4 \approx e_1 \;\textit{then}\; \mathsf{count}(e_4, y_1) \approx \mathsf{max}(0,1) \;\textit{else}\; \mathsf{count}(e_4, y_1) \approx 0\,, \\ \textit{if}\;\; e_1 \approx e_2 \;\textit{then}\; \mathsf{count}(e_1, y_2) \approx \mathsf{max}(0,1) \;\textit{else}\; \mathsf{count}(e_1, y_1) \approx 0\,, \\ \textit{if}\;\; e_2 \approx e_2 \;\textit{then}\; \mathsf{count}(e_2, y_2) \approx \mathsf{max}(0,1) \;\textit{else}\; \mathsf{count}(e_2, y_1) \approx 0\,, \\ \textit{if}\;\; e_4 \approx e_2 \;\textit{then}\; \mathsf{count}(e_4, y_2) \approx \mathsf{max}(0,1) \;\textit{else}\; \mathsf{count}(e_4, y_1) \approx 0\,, \\ u_2 \leftrightarrow y_1 \approx y_2\,, \\ u_3 \leftrightarrow \neg u_2\,, \\ u_4 \leftrightarrow (u_1 \wedge u_3)\,, \\ u_4 \end{array} \right\}.$$

Since $\rho(\varphi, \emptyset)$ is $\left(M_{\mathsf{int}} \oplus M_{\approx}^{\Omega}\right)$-unsatisfiable, it follows that φ is $(M_{\mathsf{set}} \oplus M_{\mathsf{bag}})$-unsatisfiable. $\square$

Example 38. Let $M_{\mathsf{list(set)}}$ be the model class of lists of sets, that is, $M_{\mathsf{list(set)}}$ is the same as M_{list}, except that the sort elem is renamed as set.

Next, let

$$\mathit{\Gamma}_{\mathsf{set}} = \left\{ \begin{array}{l} x_1 \approx x_2 \cup z\,, \\ z \approx \emptyset \end{array} \right\}, \qquad \mathit{\Gamma}_{\mathsf{list(set)}} = \left\{ \begin{array}{l} x_1 \approx \mathsf{car}(\ell_1)\,, \\ x_2 \approx \mathsf{car}(\ell_2)\,, \\ \ell_3 \approx \mathsf{cdr}(\ell_1)\,, \\ \ell_3 \approx \mathsf{cdr}(\ell_2)\,, \\ \ell_3 \approx \mathsf{nil}\,, \\ u \leftrightarrow \ell_1 \approx \ell_3\,, \\ u \leftrightarrow \ell_2 \approx \ell_3\,, \\ u \leftrightarrow \ell_1 \approx \ell_2\,, \\ \neg u\,, \end{array} \right\}.$$

We want to use the reduction approach in order to show that $\mathit{\Gamma}_{\mathsf{set}} \wedge \mathit{\Gamma}_{\mathsf{list(set)}}$ is $(M_{\mathsf{set}} \oplus M_{\mathsf{list(set)}})$-unsatisfiable.

Specifically, we combine the reduction functions ρ_{set} and $\rho_{\mathsf{list(set)}}$ in order to obtain a reduction function ρ from the model class $M_{\mathsf{set}} \oplus M_{\mathsf{list(set)}}$ to the model class $M_{\approx}^{\mathit{\Omega}} \oplus M_{\mathsf{cons(set)}}$, where:

- $\mathit{\Omega} = (\{\mathsf{elem}, \mathsf{set}\}, \emptyset, \{\in\})$;
- $M_{\mathsf{cons(set)}}$ is the same as M_{cons}, except that the sort elem is renamed as set.

We have

$$\rho(\mathit{\Gamma}_{\mathsf{set}} \wedge \mathit{\Gamma}_{\mathsf{list(set)}}, \emptyset) = \rho_{\mathsf{set}}(\mathit{\Gamma}_{\mathsf{set}}, \{x_1, x_2\}) \wedge \rho_{\mathsf{list(set)}}(\mathit{\Gamma}_{\mathsf{list(set)}}, \{x_1, x_2\})\,.$$

In particular, we have

$$\rho_{\mathsf{set}}(\mathit{\Gamma}_{\mathsf{set}}, \{x_1, x_2\}) = \left\{ \begin{array}{l} e \in x_1 \;\leftrightarrow\; (e \in x_2 \;\vee e \in z)\,, \\ e \notin z\,, \\ x_1 \not\approx x_2 \;\rightarrow\; ((e \in x_1 \;\wedge\; e \notin x_2) \;\vee\; (e \notin x_1 \;\wedge\; e \in x_2)) \end{array} \right\}$$

and

$$\rho_{\mathsf{list(set)}}(\mathit{\Gamma}_{\mathsf{list(set)}}, \{x_1, x_2\}) = \left\{ \begin{array}{l} \ell_1 \not\approx \mathsf{nil} \;\rightarrow\; \ell_1 \approx \mathsf{cons}(x_1, \ell_1')\,, \\ \ell_2 \not\approx \mathsf{nil} \;\rightarrow\; \ell_2 \approx \mathsf{cons}(x_2, \ell_2')\,, \\ \ell_1 \not\approx \mathsf{nil} \;\rightarrow\; \ell_1 \approx \mathsf{cons}(x_1', \ell_3)\,, \\ \ell_2 \not\approx \mathsf{nil} \;\rightarrow\; \ell_2 \approx \mathsf{cons}(x_2', \ell_3)\,, \\ \ell_3 \approx \mathsf{nil}\,, \\ u \leftrightarrow \ell_1 \approx \ell_3\,, \\ u \leftrightarrow \ell_2 \approx \ell_3\,, \\ u \leftrightarrow \ell_1 \approx \ell_2\,, \\ \neg u\,, \end{array} \right\}.$$

Since $\rho(\varphi, \emptyset)$ is $\big(M_{\approx}^{\mathit{\Omega}} \oplus M_{\mathsf{cons(set)}}\big)$-unsatisfiable, it follows that φ is $(M_{\mathsf{set}} \oplus M_{\mathsf{list(set)}})$-unsatisfiable.

Finally, note that $\rho_{\mathsf{set}}(\mathit{\Gamma}_{\mathsf{set}}, \emptyset) \wedge \rho_{\mathsf{list(set)}}(\mathit{\Gamma}_{\mathsf{list(set)}}, \emptyset)$ is $\big(M_{\approx}^{\mathit{\Omega}} \oplus M_{\mathsf{cons(set)}}\big)$-satisfiable. Consequently, this example shows the *raison d'être* of the second argument of reduction functions. □

12.2 The proof

Theorem 39. *Let us be given the following model classes:*

$$M_1 = (\Sigma_1, \mathbf{A}_1)\,, \qquad N_1 = (\Omega_1, \mathbf{B}_1)\,,$$
$$M_2 = (\Sigma_2, \mathbf{A}_2)\,, \qquad N_2 = (\Omega_2, \mathbf{B}_2)\,.$$

Assume that:

- $\Sigma_1^{\mathrm{F}} \cap \Sigma_2^{\mathrm{F}} = \emptyset$*;*
- $\Sigma_1^{\mathrm{P}} \cap \Sigma_2^{\mathrm{P}} = \emptyset$*;*
- $\Omega_1 \subseteq \Sigma_1$ *and* $\Omega_1^{\mathrm{S}} = \Sigma_1^{\mathrm{S}}$*;*
- $\Omega_2 \subseteq \Sigma_2$ *and* $\Omega_2^{\mathrm{S}} = \Sigma_2^{\mathrm{S}}$*;*
- ρ_1 *is a reduction function from* M_1 *to* N_1*;*
- ρ_2 *is a reduction function from* M_2 *to* N_2*.*

Then $M_1 \oplus M_2$ *effectively reduces to* $N_1 \oplus N_2$*.* □

Proof. Let φ be a $(\Sigma_1 \cup \Sigma_2)$-formula, let $X_0 \subseteq \mathit{free}(\varphi)$, and let π be a (Σ_1, Σ_2)-purification function. Also let:

$$\begin{aligned}
\pi(\varphi) &= \varphi_1 \wedge \varphi_2\,,\\
X &= \mathit{free}(\varphi)\,,\\
X_1 &= \mathit{free}(\varphi_1)\,,\\
X_2 &= \mathit{free}(\varphi_2)\,,\\
X_1' &= X_1 \cap (X_0 \cup X_2)\,,\\
X_2' &= X_2 \cap (X_0 \cup X_1)\,,\\
\psi_1 &= \rho_1(\varphi_1, X_1')\,,\\
\psi_2 &= \rho_2(\varphi_2, X_2')\,,\\
Y_1 &= \mathit{free}(\psi_1)\,,\\
Y_2 &= \mathit{free}(\psi_2)\,.
\end{aligned}$$

Without loss of generality, assume that $(Y_1 \setminus X_1) \cap (Y_2 \setminus X_2) = \emptyset$.[8] Also, note that we have $X_0 \subseteq X \subseteq X_1 \cup X_2 \subseteq Y_1 \cup Y_2$.

Consider the function ρ defined by letting

$$\rho(\varphi, X_0) = \psi_1 \wedge \psi_2\,.$$

We claim that ρ is a reduction function from $M_1 \oplus M_2$ to $N_1 \oplus N_2$. We prove this claim by showing that ρ satisfies properties (a) and (b) of Definition 28.

[8] Note that this is possible because the free constant symbols in $Y_i \setminus X_i$ have been introduced by an application of the reduction function ρ_i, and we can assume that the reduction functions ρ_i have separately introduced disjoint sets of free constant symbols.

(a) Assume that

$$\mathcal{A} \models_{M_1 \oplus M_2} \varphi\,,$$

for some $(\Sigma_1 \cup \Sigma_2)$-structure over X. By Definition 26, it follows that there exists a $(\Sigma_1 \cup \Sigma_2)$-structure $\mathcal{B}$ over $X_1 \cup X_2$ such that

$$\mathcal{B} \models_{M_1 \oplus M_2} \varphi_1 \wedge \varphi_2\,,$$
$$\mathcal{B}^{\Sigma_1 \cup \Sigma_2, X} \cong \mathcal{A}\,,$$

which implies

$$\mathcal{B}^{\Sigma_1, X_1} \models_{M_1} \varphi_1\,,$$
$$\mathcal{B}^{\Sigma_2, X_2} \models_{M_2} \varphi_2\,.$$

By Definition 28, it follows that there exist an Ω_1-structure $\mathcal{C}$ over Y_1, and an Ω_2-structure $\mathcal{D}$ over Y_2 suth that

$$\mathcal{C} \models_{N_1} \psi_1\,,$$
$$\mathcal{D} \models_{N_2} \psi_2\,,$$
$$\mathcal{C}^{\Omega_1, X_1} \cong \mathcal{B}^{\Omega_1, X_1}\,,$$
$$\mathcal{D}^{\Omega_2, X_2} \cong \mathcal{B}^{\Omega_2, X_2}\,.$$

By noting that

$$Y_1 \cap Y_2 = X_1 \cap X_2\,,$$

it follows that

$$\mathcal{C}^{\Omega_1 \cap \Omega_2, Y_1 \cap Y_2} \cong \mathcal{D}^{\Omega_1 \cap \Omega_2, Y_1 \cap Y_2}\,.$$

By Theorem 22, there exists an $(\Omega_1 \cup \Omega_2)$-structure $\mathcal{E}$ over $Y_1 \cup Y_2$ such that

$$\mathcal{E} \models_{N_1 \oplus N_2} \psi_1 \wedge \psi_2\,,$$
$$\mathcal{E}^{\Omega_1, Y_1} \cong \mathcal{C}\,,$$
$$\mathcal{E}^{\Omega_2, Y_2} \cong \mathcal{D}\,.$$

By noting that

$$X \subseteq Y_1 \cup Y_2\,,$$

it follows that

$$\mathcal{E}^{\Omega_1 \cup \Omega_2, X} \cong \mathcal{A}^{\Omega_1 \cup \Omega_2, X}\,.$$

(b) Assume that

$$\mathcal{B} \models_{N_1 \oplus N_2} \psi_1 \wedge \psi_2\,,$$

where $\mathcal{B}$ is a $(\Omega_1 \cup \Omega_2)$-structure over $Y_1 \cup Y_2$. It follows that

$$\mathcal{B}^{\Omega_1, Y_1} \models_{N_1} \psi_1 \,,$$
$$\mathcal{B}^{\Omega_2, Y_2} \models_{N_2} \psi_2 \,.$$

By Definition 28, there exist a Σ_1-structure $\mathcal{C}$ over X_1, and a Σ_2-structure $\mathcal{D}$ over X_2 such that

$$\mathcal{C} \models_{M_1} \varphi_1 \,,$$
$$\mathcal{C}^{\Sigma_1^{\mathrm{S}}, X_1'} \cong \mathcal{B}^{\Sigma_1^{\mathrm{S}}, X_1'} \,,$$

and

$$\mathcal{D} \models_{M_2} \varphi_2 \,,$$
$$\mathcal{D}^{\Sigma_2^{\mathrm{S}}, X_2'} \cong \mathcal{B}^{\Sigma_2^{\mathrm{S}}, X_2'} \,.$$

But then, by observing that

$$X_1 \cap X_2 \subseteq X_1' \cap X_2' \,,$$
$$\Sigma_1^{\mathrm{F}} \cap \Sigma_2^{\mathrm{F}} = \emptyset \,,$$
$$\Sigma_1^{\mathrm{P}} \cap \Sigma_2^{\mathrm{P}} = \emptyset \,,$$

we obtain

$$\mathcal{C}^{\Sigma_1 \cap \Sigma_2, X_1 \cap X_2} \cong \mathcal{D}^{\Sigma_1 \cap \Sigma_2, X_1 \cap X_2} \,.$$

By Theorem 22, there exists a $(\Sigma_1 \cup \Sigma_2)$-structure $\mathcal{E}$ over $X_1 \cup X_2$ such that

$$\mathcal{E} \models_{M_1 \oplus M_2} \varphi_1 \wedge \varphi_2 \,,$$
$$\mathcal{E}^{\Sigma_1, X_1} \cong \mathcal{C} \,,$$
$$\mathcal{E}^{\Sigma_2, X_2} \cong \mathcal{D} \,.$$

By noting that

$$X_0 \subseteq X_1' \cup X_2' \,,$$
$$X_1' \subseteq X_1 \,,$$
$$X_2' \subseteq X_2 \,,$$

it follows that

$$\mathcal{E}^{\Sigma_1^{\mathrm{S}} \cup \Sigma_2^{\mathrm{S}}, X_0} \cong \mathcal{B}^{\Sigma_1^{\mathrm{S}} \cup \Sigma_2^{\mathrm{S}}, X_0} \,.$$

Finally, by Definition 26, we also have that

$$\mathcal{E}^{\Sigma_1 \cup \Sigma_2, X} \models_{M_1 \oplus M_2} \varphi \,. \qquad \blacksquare$$

13 Conclusion

We presented an approach for designing decision procedures based on the reduction of complex model classes to simpler ones.

Given a Σ-model class M and an Ω-model class N, we use a *reduction function* from M to N in order to translate a Σ-formula φ into an Ω-formula ψ such that φ is M-satisfiable if and only if ψ is N-satisfiable.

We used reduction functions in order to show that:

– the model class M_{list} of lists reduces to the model class M_{cons} of constructors;
– the model class M_{array} of arrays reduces to the model class $M_{\approx}$ of equality;
– the model class M_{set} of sets reduces to the model class $M_{\approx}$ of equality;
– the model class M_{bag} of multisets reduces to the model class M_{int} of integers extended with uninterpreted function symbols.

Finally, we provided a method for combining reduction functions. More precisely, assume that ρ_i is a reduction function from M_i to N_i, for $i = 1, 2$, and that the signatures of M_1 and M_2 do not share any function or predicate symbols. We showed how to use ρ_1 and ρ_2 as black boxes in order to construct a reduction function ρ from $M_1 \oplus M_2$ to $N_1 \oplus N_2$.

References

1. Wilhelm Ackermann. *Solvable Cases of the Decision Problem.* North-Holland, 1954.
2. Pascal Fontaine, Silvio Ranise, and Calogero G. Zarba. Combining lists with non-stably infinite theories. In Franz Baader and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 3452 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2005.
3. Jürgen Giesl and Deepak Kapur. Deciding inductive validity of equations. In Franz Baader, editor, *Automated Deduction – CADE-19*, volume 2741 of *Lecture Notes in Computer Science*, pages 17–31. Springer, 2003.
4. Deepak Kapur. Rewriting, induction and decision procedures: A case study of Presburger arithmetic. In Götz Alefeld, Jiri Rohn, Siegfried Rump, and Tetsuro Yamamato, editors, *Symbolic-algebraic Methods and Verification Methods*, pages 129–144. Springer, 2001.
5. Deepak Kapur and Hanthao Zhang. An overview of Rewrite Rule Laboratory. *Journal of Computer and Mathematics with Applications*, 29(2):91–114, 1995.
6. William Pugh. The Omega test: A fast integer programming algorithm for dependence analysis. In *Supercomputing*, pages 4–13, 1991.
7. Silvio Ranise, Christophe Ringeissen, and Calogero G. Zarba. Combining data structures with nonstably infinite theories using many-sorted logic. In Bernhard Gramlich, editor, *Frontiers of Combining Systems*, volume 3717 of *Lecture Notes in Computer Science*, pages 48–64. Springer, 2005.
8. Christophe Ringeissen. Cooperation of decision procedures for the satisfiability problem. In Franz Baader and Klaus U. Schulz, editors, *Frontiers of Combining Systems*, volume 3 of *Applied Logic Series*, pages 121–140. Kluwer, 1996.

9. Aaron Stump, Clark W. Barrett, David L. Dill, and Jeremy R. Levitt. A decision procedure for an extensional theory of arrays. In *16th Annual IEEE Symposium on Logic in Computer Science*, pages 29–37. IEEE Computer Society, 2001.
10. Cesare Tinelli and Mehdi T. Harandi. A new correctness proof of the Nelson-Oppen combination procedure. In Franz Baader and Klaus U. Schulz, editors, *Frontiers of Combining Systems*, volume 3 of *Applied Logic Series*, pages 103–120. Kluwer, 1996.
11. Calogero G. Zarba. Combining multisets with integers. In Andrei Voronkov, editor, *Automated Deduction – CADE-18*, volume 2392 of *Lecture Notes in Computer Science*, pages 363–376. Springer, 2002.
12. Calogero G. Zarba. Combining sets with elements. In Nachum Dershowitz, editor, *Verification: Theory and Practice*, volume 2772 of *Lecture Notes in Computer Science*, pages 762–782. Springer, 2004.