Axiomatizations and Computability of Weighted Monadic Second-Order Logic

Antonis Achilleos

Reykjavík University, Iceland

Mathias Ruggaard Pedersen

Reykjavík University, Iceland

Abstract

Weighted monadic second-order logic is a weighted extension of monadic second-order logic which captures exactly the behaviour of weighted automata. We give complete axiomatizations for most parts of this logic, and give evidence that a complete axiomatization of the full logic is unlikely. Furthermore, we discuss how common decision problems for logical languages can be adapted to the weighted setting, and show that many of these are decidable, though they inherit bad complexity from the underlying first- and second-order logics. However, we show that a weighted adaptation of satisfiability is undecidable for the full logic.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Proof theory; Theory of computation → Finite model theory

Keywords and phrases Weighted Logic, Monadic Second-Order Logic, Axiomatization, Weighted Automata, Satisfiability

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding This work has been funded by the project "Open Problems in the Equational Logic of Processes (OPEL)" (grant no. 196050-051) and the project "Epistemic Logic for Distributed Runtime Monitoring" (grant no. 184940-052) of the Icelandic Research Fund.

1 Introduction

Weighted logics are a quantitative generalization of classical logics that allows one to reason about quantities such as probabilities, cost, production, or energy consumption in systems [2,5,10]. These kinds of logic are important, since they allow us to describe not only that e.g. a certain task was completed, but also that only a specific amount of resources were consumed in order to complete the task. One of the main results of the theory of weighted logics is the correspondence between weighted monadic second-order logic [2] or quantitative monadic second-order logic [10] and weighted automata, thus generalizing the classical result of Büchi, Elgot, and Trakhtenbrot [1,3,20], of the equivalence between classical finite automata and monadic second-order logic (MSO). This is important because it shows that weighted or quantitative MSO are well-suited to reason about weighted automata, which themselves are a popular tool for modeling systems, having found applications in areas such as image compression [8] and natural language processing [9,16]. The correspondence between weighted MSO and weighted automata has been adapted to many other computational models, such as weighted Muller tree automata [17] and weighted picture automata [4].

Complete axiomatizations for weighted logics as well as their decision problems have been well-studied in the context of weighted extensions of modal logics for weighted transition systems. Larsen and Mardare [11] gave a complete axiomatization for weighted modal logic on weighted transition systems, and they later extended this work to also consider concurrency [13]. Hansen et al. gave complete axiomatization for a logic to reason about bounds in weighted transitions systems in [7], where they also show the decidability of the satisfiability problem. Larsen et al. proved in [14] that the satisfiability problem is decidable for weighted logic with recursion, in which recursive equations can describe infinite behaviour. Similarly, Larsen et al. gave in [12] both a complete axiomatization and a decision procedure for satisfiability for the alternation-free fragment of a weighted extension of the μ-calculus on weighted transition systems.

However, while MSO and first-order logic (FO) have been well-studied for decades, there has not been a study of its weighted extensions at the level of logic. In this paper we initiate this study by giving axiomatisations of fragments of weighted MSO and considering the decidability of some of its decision problems. We give a complete axiomatization of the full second syntactic layer of weighted MSO and a complete axiomatization for a fragment of the third and final syntactic layer of weighted MSO. We also show that the model checking, satisfiability, and validity problems, appropriately translated to the weighted setting, are decidable for the second layer of the logic, although these inherit the PSPACE-completeness and non-elementary complexity from MSO for model checking and, respectively, for satisfiability and validity. However, for the third layer of the logic, things are more complicated. The model checking problem remains decidable, but we show that the satisfiability problem is undecidable, even for the first-order fragment. We also conjecture that the validity problem is undecidable, and we give some evidence to suggest this through examples of undecidability in terms of concrete semantics. The possible undecidability of the validity problem is of particular interest, since this result will imply that the full weighted MSO has no recursive and complete axiomatization.

Omitted proofs can be found in the appendix.

2 Preliminaries

Denote by $\mathbb{N}\{\!|X|\!\}$ the collection of all finite multisets over X, where a finite multiset is a function $f : X \to \mathbb{N}$ such that $f(x) \neq 0$ for finitely many $x \in X$. Intuitively, $f(x)$ tells us how many times the element x occurs in the multiset $\mathbb{N}\{\!|X|\!\}$. We will use $\{\!| \cdot |\!\}$ to denote a multiset, so that e.g. $\{\!|1, 1, 2, 3|\!\}$ is the multiset that contains two 1's, one 2, and one 3. The union $\uplus$ of two multisets f and g is defined pointwise as $(f \uplus g)(x) = f(x) + g(x)$.

A semiring is a tuple $(X, +, \times, 0, 1)$ such that $(X, \times, 1)$ is a monoid ($\times$ is an associative binary operation on X, with 1 as an identity element), $(X, +, 0)$ is a commutative monoid (it is a monoid and $+$ is commutative), $\times$ distributes over $+$, and $0 \times x = x \times 0 = 0$ for all $x \in X$. Some common examples of semirings are $(\mathbb{Z}, +, \times, 0, 1)$, the integers with the usual sum and product, and $(\{0, 1\}, \vee, \wedge, 0, 1)$, the Boolean semiring with the usual Boolean disjunction and conjunction. For our purposes, another important example of semirings is that of $(\mathbb{N}\{\!|X^*|\!\}, \uplus, \cdot, \emptyset, \{\!|\varepsilon|\!\})$, the semiring over multisets of sequences over X, with multiset union as sum, concatenation defined as $\{\!|w|\!\} \cdot \{\!|w'|\!\} = \{\!|ww'|\!\}$ as product, the empty set as zero, and the multiset containing only the empty string as identity.

3 Syntax and semantics

Our presentation of weighted MSO and FO follows the style of [5], in which the logic is separated into three different layers. The first layer is simply MSO or FO; the second layer is built from single values and if-then-else statements with MSO or FO formulas as conditions; whereas the third and last layer allows one to combine the single values from the second layer into more complex expressions using sums and products.

We use a countably infinite set of first-order variables $\mathcal{V}_{FO}$, a countably infinite set of second-order variables $\mathcal{V}_{SO}$, a finite alphabet Σ, and an arbitrary set R of weights. The syntax of weighted MSO is given by the following grammar.

$$\varphi ::= \top \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \forall x.\varphi \mid \forall X.\varphi \qquad \text{(MSO)}$$
$$\Psi ::= r \mid \varphi \mathbin{?} \Psi_1 : \Psi_2 \qquad \text{(step-wMSO)}$$
$$\Phi ::= \mathbf{0} \mid \textstyle\prod_x \Psi \mid \varphi \mathbin{?} \Phi_1 : \Phi_2 \mid \Phi_1 + \Phi_2 \mid \sum_x \Phi \mid \sum_X \Phi \qquad \text{(core-wMSO)}$$

where $a \in \Sigma$, $r \in R$, $x, y \in \mathcal{V}_{FO}$, and $X \in \mathcal{V}_{SO}$. In the rest of the paper, we use φ to denote MSO formulas, Ψ to denote step-wMSO formulas, Φ to denote core-wMSO formulas, and χ to denote step-wMSO or core-wMSO formulas.

In a similar fashion, we obtain step-wFO by only allowing conditioning on first-order formulas in step-wMSO and core-wFO by only allowing first-order formulas and removing the construct $\sum_X \Phi$ which sums over a second-order variable.

The formulas φ of MSO are interpreted over words $w \in \Sigma^+$ together with a valuation σ of this word, which assigns to each first-order variable a position in the word and to each second-order variable a set of positions in the word.

When interpreted on a string, a formula outputs a value, which, concretely, may be a single weight, a sequence, or, say, a set (or multiset) of more elementary values. To preserve the generality of the logic, the semantics are given in two steps. The first is an abstract semantics, where the meaning of a formula is given as a multiset of sequences of weights. The second is a concrete semantics, where one can translate the abstract semantics into a given semiring structure, by assuming an appropriate operator on the abstract values.

We denote by Σ^+_σ the set of pairs (w, σ) where $w \in \Sigma^+$ and σ is a valuation of w. Let x be a first-order (respectively, let X be a second-order) variable and $i \in \{1, \dots, |w|\}$ (respectively, $I \subseteq \{1, \dots, |w|\}$). By $\sigma[x \mapsto i]$ (respectively $\sigma[X \mapsto I]$) we denote the valuation that maps each variable y and Y to $\sigma(y)$ and $\sigma(Y)$, if $y \neq x$ (respectively, if $Y \neq X$), and x to i (respectively, X to I). The semantics of MSO on finite words is standard and can be found in e.g. [15]. In this paper, (w, σ) will always be a pair from Σ^+_σ.

We denote by $\llbracket \varphi \rrbracket$ the set of all pairs $(w, \sigma) \in \Sigma^+_\sigma$ that satisfy φ. Likewise, for a set Γ of MSO formulas, we define $\llbracket \Gamma \rrbracket$ by $\llbracket \Gamma \rrbracket = \Sigma^+_\sigma$ if $\Gamma = \emptyset$ and $\llbracket \Gamma \rrbracket = \bigcap_{\varphi \in \Gamma} \llbracket \varphi \rrbracket$ otherwise. The semantics of formulas Ψ of step-wMSO is given by a function $\llbracket \cdot \rrbracket : \Sigma^+_\sigma \to R$, defined by

$$\llbracket r \rrbracket (w, \sigma) = r \quad \text{and} \quad \llbracket \varphi \mathbin{?} \Psi_1 : \Psi_2 \rrbracket (w, \sigma) = \begin{cases} \llbracket \Psi_1 \rrbracket (w, \sigma) \text{ if } (w, \sigma) \models \varphi \\ \llbracket \Psi_2 \rrbracket (w, \sigma) \text{ otherwise.} \end{cases}$$

The semantics of formulas Φ of core-wMSO is given by the function $\llbracket \cdot \rrbracket : \Sigma^+_\sigma \to \mathbb{N}\{\!|R^*|\!\}$:

$$\begin{aligned}
\llbracket \mathbf{0} \rrbracket (w, \sigma) &= \emptyset \\
\llbracket \textstyle\prod_x \Psi \rrbracket (w, \sigma) &= \{\!| r_1 r_2 \dots r_{|w|} |\!\} \text{ where } r_i = \llbracket \Psi \rrbracket (w, \sigma[x \mapsto i]) \text{ for } 1 \leq i \leq |w| \\
\llbracket \varphi \mathbin{?} \Phi_1 : \Phi_2 \rrbracket (w, \sigma) &= \begin{cases} \llbracket \Phi_1 \rrbracket (w, \sigma) \text{ if } (w, \sigma) \models \varphi \\ \llbracket \Phi_2 \rrbracket (w, \sigma) \text{ otherwise} \end{cases} \\
\llbracket \Phi_1 + \Phi_2 \rrbracket (w, \sigma) &= \llbracket \Phi_1 \rrbracket (w, \sigma) \uplus \llbracket \Phi_2 \rrbracket (w, \sigma) \\
\llbracket \textstyle\sum_x \Phi \rrbracket (w, \sigma) &= \biguplus_{i \in \{1, \dots, |w|\}} \llbracket \Phi \rrbracket (w, \sigma[x \mapsto i]) \\
\llbracket \textstyle\sum_X \Phi \rrbracket (w, \sigma) &= \biguplus_{I \subseteq \{1, \dots, |w|\}} \llbracket \Phi \rrbracket (w, \sigma[X \mapsto I])
\end{aligned}$$

Let Γ be a set of MSO formulas. We say that two formulas χ_1 and χ_2 are semantically Γ-equivalent and write $\chi_1 \sim_\Gamma \chi_2$ if $\llbracket \chi_1 \rrbracket (w, \sigma) = \llbracket \chi_2 \rrbracket (w, \sigma)$ for all $(w, \sigma) \in \llbracket \Gamma \rrbracket$. If $\Gamma = \emptyset$, we simply write $\chi_1 \sim \chi_2$ and say that χ_1 and χ_2 are semantically equivalent.

Concrete semantics To obtain the concrete semantics of a formula for a given semiring structure $(X, +, \times, 0, 1)$, we define an *aggregation function* $\texttt{aggr} : \mathbb{N}\{\!|R^*|\!\} \to X$. Note that the set X may be different from the set of weights R.

▶ **Example 1.** Let $\Sigma = \{a, b\}$, $R = \{0, 1\}$ and consider the max-plus semiring $(\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$. We wish to count the maximum number of consecutive a's in a given string $w \in \Sigma^*$. We define the aggregation function as $\texttt{aggr}(M) = \max_{r_1 \dots r_n \in M}(r_1 + \cdots + r_n)$, thus interpreting the sum and product of the multiset sequence semiring ($\uplus$ and $\cdot$) as the corresponding sum and product ($\max$ and $+$) in the max-plus semiring. Now define the first-order formulas φ as $\varphi = x \le y \wedge \forall z.((x \le z \wedge z \le y) \to P_a(z))$, and let $\Psi = \varphi \,?\, 1 : 0$, $\Phi' = \prod_y \Psi$, and $\Phi = \sum_x \Phi'$, so that $\Phi = \sum_x \prod_y \varphi \,?\, 1 : 0$. Consider the string $w = abaa$, which has a maximum number of two consecutive a's. We find that

$$
\begin{aligned}
[\![\Phi]\!]\,(w,\sigma) &= [\![\Phi']\!]\,(w,\sigma[x{\mapsto}1]) \uplus [\![\Phi']\!]\,(w,\sigma[x{\mapsto}2]) \uplus [\![\Phi']\!]\,(w,\sigma[x{\mapsto}3]) \uplus [\![\Phi']\!]\,(w,\sigma[x{\mapsto}4]) \\
&= [\![\Psi]\!]\,(w,\sigma[x{\mapsto}1,y{\mapsto}1])\,[\![\Psi]\!]\,(w,\sigma[x{\mapsto}1,y{\mapsto}2])\,[\![\Psi]\!]\,(w,\sigma[x{\mapsto}1,y{\mapsto}3]) \\
&\quad\ [\![\Psi]\!]\,(w,\sigma[x{\mapsto}1,y{\mapsto}4]) \\
&\qquad \uplus [\![\Phi']\!]\,(w,\sigma[x{\mapsto}2]) \uplus [\![\Phi']\!]\,(w,\sigma[x{\mapsto}3]) \uplus [\![\Phi']\!]\,(w,\sigma[x{\mapsto}4]) \\
&= \{\!|1000|\!\} \uplus [\![\Phi']\!]\,(w,\sigma[x{\mapsto}2]) \uplus [\![\Phi']\!]\,(w,\sigma[x{\mapsto}3]) \uplus [\![\Phi']\!]\,(w,\sigma[x{\mapsto}4]) \\
&= \{\!|1000|\!\} \uplus \{\!|0000|\!\} \uplus \{\!|0011|\!\} \uplus \{\!|0001|\!\} = \{\!|1000, 0000, 0011, 0001|\!\}
\end{aligned}
$$

and hence the concrete semantics become

$$
\begin{aligned}
\texttt{aggr}([\![\Phi]\!]\,(w,\sigma)) &= \texttt{aggr}(\{\!|1000, 0000, 0011, 0001|\!\}) \\
&= \max\{1+0+0+0, 0+0+0+0, 0+0+1+1, 0+0+0+1\} \\
&= \max\{1, 0, 2, 1\} = 2,
\end{aligned}
$$

which is exactly the maximum number of two consecutive a's in w.

4 Axioms

Just as the syntax of the logic was given in three layers, we also present the axioms of the logic in three layers, one for each of the syntactic layers. Our axiomatization of core-wMSO will only be for the fragment that excludes the general sum construct. Indeed, in Section 6 we present reasons to suspect that the full core-wMSO has no recursive and complete axiomatization. We note that the proofs of completeness do not rely on any properties of MSO itself, apart from it having a complete axiomatization, and therefore the axiomatizations also apply to step-wFO and core-wFO.

For the step-wMSO and core-wMSO layers, which are not Boolean, we give an axiomatization in terms of equational logic. For a set Γ of MSO formulas, we use the notation $\Gamma \vdash \chi_1 \approx \chi_2$ to mean that under the assumptions in Γ, χ_1 is equivalent to χ_2. These equations must satisfy the axioms of equational logic, which are reflexivity, symmetry, transitivity, and congruence, as reported in Table 1. Note that the congruence rule for sum, cong+, only applies to the core-wMSO layer, since step-wMSO has no sum operator. Furthermore, the congruence rule for the conditional operator, cong?, is not strictly necessary to include, since it can be derived from the axioms that we introduce later. However, to follow standard presentations of equational logic, we include it as part of the axioms here.

(ref):	$\Gamma \vdash \chi \approx \chi$
(sym):	$\Gamma \vdash \chi_1 \approx \chi_2$ implies $\Gamma \vdash \chi_2 \approx \chi_1$
(trans):	$\Gamma \vdash \chi_1 \approx \chi_2$ and $\Gamma \vdash \chi_2 \approx \chi_3$ implies $\Gamma \vdash \chi_1 \approx \chi_3$
(cong?):	$\Gamma \vdash \chi_1 \approx \chi_1'$ and $\Gamma \vdash \chi_2 \approx \chi_2'$ implies $\Gamma \vdash \varphi \,?\, \chi_1 : \chi_2 \approx \varphi \,?\, \chi_1' : \chi_2'$
(cong+):	$\Gamma \vdash \chi_1 \approx \chi_1'$ and $\Gamma \vdash \chi_2 \approx \chi_2'$ implies $\Gamma \vdash \chi_1 + \chi_2 \approx \chi_1' + \chi_2'$

Table 1 Axioms for equational logic.

$(S1)$:	$\Gamma \vdash \Psi_1 \approx \Psi_2$ implies $\Gamma \cup \{\varphi\} \vdash \Psi_1 \approx \Psi_2$	for any MSO formula φ
$(S2)$:	$\Gamma \vdash \neg\varphi \,?\, \Psi_1 : \Psi_2 \approx \varphi \,?\, \Psi_2 : \Psi_1$	
$(S3)$:	$\Gamma \vdash \varphi \,?\, \Psi_1 : \Psi_2 \approx \Psi_1$	if $\Gamma \vdash \varphi$
$(S4)$:	if $\Gamma \cup \{\varphi\} \vdash \Psi_1 \approx \Psi,$ and $\Gamma \cup \{\neg\varphi\} \vdash \Psi_2 \approx \Psi,$ then $\Gamma \vdash \varphi \,?\, \Psi_1 : \Psi_2 \approx \Psi$	

Table 2 Axioms for step-wMSO.

4.1 MSO

MSO over finite strings is equivalent to finite automata [1, 3, 20], and therefore it also has a decidable validity problem (albeit with a nonelementary complexity). This means that the theory of MSO over finite strings has a recursive and complete axiomatization. One such axiomatization is given in [6], and therefore for a set $\Gamma \cup \{\varphi\}$ of MSO formulas, $\Gamma \vdash \varphi$ means that φ is derivable from these axioms and Γ (Γ may be omitted when empty). Since FO over finite strings also has a decidable validity problem, it likewise has a recursive and complete axiomatization. For the purpose of this paper, we fix one such axiomatization, and we can thus also write $\Gamma \vdash \varphi$ when $\Gamma \cup \{\varphi\}$ is a set of FO formulas.

▶ **Theorem 2** (Completeness of MSO [6]). *$\models \varphi$ if and only if $\vdash \varphi$.*

▶ **Corollary 3.** *For finite Γ we have that $\Gamma \models \varphi$ if and only if $\Gamma \vdash \varphi$.*

4.2 step-wMSO

The equational axioms for step-wMSO are given in Table 2. Axiom $(S1)$ allows one to add additional assumptions to Γ, and $(S2)$ shows how negation affects the conditional operator by switching the order of the results. Axiom $(S3)$ shows that if the formula φ that is being conditioned on can be derived from Γ itself, then the first choice of the conditional will always be taken. Finally, $(S4)$ gives a way to remove assumptions and put them into a conditional statement instead: If the first choice of the conditional is equivalent to Ψ under the assumption that φ is true, and the second choice of the conditional is equivalent to Ψ under the assumption that φ is false, then the conditional is equivalent to Ψ.

Before proving that the axioms given in Table 2 are complete, we first give some examples of theorems that can be derived from the axioms, some of which will be used in the proof of completeness. The first two of these are particularly interesting, since they give properties that are common in many logical systems, namely the principle of explosion and the cut elimination rule. The remaining theorems show that the conditional operator behaves as expected, and that all of these behaviours can be inferred from the four axioms of Table 2.

▶ **Proposition 4.** *The following theorems can be derived in step-wMSO.*

1. *$\Gamma \vdash \Psi_1 \approx \Psi_2$ for any Ψ_1 and Ψ_2 if Γ is inconsistent.*
2. *$\Gamma \vdash \Psi_1 \approx \Psi_2$ if $\Gamma \vdash \varphi$ and $\Gamma \cup \{\varphi\} \vdash \Psi_1 \approx \Psi_2$.*

3. $\Gamma \vdash \varphi \mathbin{?} \Psi : \Psi \approx \Psi$.
4. *If* $\Gamma \cup \{\varphi_1, \varphi_2\} \vdash \Psi_1 \approx \Psi_1'$, $\Gamma \cup \{\varphi_1, \neg\varphi_2\} \vdash \Psi_1 \approx \Psi_2'$, $\Gamma \cup \{\neg\varphi_1, \varphi_2\} \vdash \Psi_2 \approx \Psi_1'$, *and* $\Gamma \cup \{\neg\varphi_1, \neg\varphi_2\} \vdash \Psi_2 \approx \Psi_2'$, *then* $\Gamma \vdash \varphi_1 \mathbin{?} \Psi_1 : \Psi_2 \approx \varphi_2 \mathbin{?} \Psi_1' : \Psi_2'$.
5. $\Gamma \vdash \varphi_1 \mathbin{?} \Psi_1 : \Psi_2 \approx \varphi_2 \mathbin{?} \Psi_1 : \Psi_2$ *if* $\Gamma \vdash \varphi_1 \leftrightarrow \varphi_2$.
6. $\Gamma \vdash \varphi \mathbin{?} \Psi_1 : \Psi_2 \approx \Psi_2$ *if* $\Gamma \vdash \neg\varphi$.
7. *If* $\Gamma \cup \{\varphi\} \vdash \Psi_1 \approx \Psi_1'$ *and* $\Gamma \cup \{\neg\varphi\} \vdash \Psi_2 \approx \Psi_2'$ *then* $\Gamma \vdash \varphi \mathbin{?} \Psi_1 : \Psi_2 \approx \varphi \mathbin{?} \Psi_1' : \Psi_2'$.
8. *If* $\Gamma \cup \{\varphi\} \vdash \Psi_1 \approx \Psi_2$ *and* $\Gamma \cup \{\neg\varphi\} \vdash \Psi_1 \approx \Psi_2$ *then* $\Gamma \vdash \Psi_1 \approx \Psi_2$.
9. $\Gamma \cup \{\varphi\} \vdash \varphi \mathbin{?} \Psi_1 : \Psi_2 \approx \Psi_1$.

The proof of completeness is by a case analysis and induction on the structure of the two formulas Ψ_1 and Ψ_2. Lemma 5 covers the case where both sides of the equation are conditional statements.

▶ **Lemma 5.** *If* $\varphi_1 \mathbin{?} \Psi_1' : \Psi_1'' \sim_\Gamma \varphi_2 \mathbin{?} \Psi_2' : \Psi_2''$, *then*

$$\Psi_1' \sim_{\Gamma\cup\{\varphi_1,\varphi_2\}} \Psi_2', \quad \Psi_1' \sim_{\Gamma\cup\{\varphi_1,\neg\varphi_2\}} \Psi_2'', \quad \Psi_1'' \sim_{\Gamma\cup\{\neg\varphi_1,\varphi_2\}} \Psi_2', \text{ and } \Psi_1'' \sim_{\Gamma\cup\{\neg\varphi_1,\neg\varphi_2\}} \Psi_2''.$$

▶ **Theorem 6.** *For finite* Γ *we have* $\Psi_1 \sim_\Gamma \Psi_2$ *if and only if* $\Gamma \vdash \Psi_1 \approx \Psi_2$.

Proof sketch. If Γ is inconsistent, we are done by Proposition 4(1), which uses axioms $(S2)$ and $(S3)$. If Γ is consistent, the proof proceeds by induction on the maximum syntactic depth of Ψ_1 and Ψ_2. The base case is taken care of by reflexivity, since in this case we must have $\Psi_1 = \Psi_2 = r$. For the inductive step, we have three cases to consider: (1) $\Psi_1 = \varphi \mathbin{?} \Psi_1' : \Psi_1''$ and $\Psi_2 = r_2$, (2) $\Psi_1 = r_1$ and $\Psi_2 = \varphi \mathbin{?} \Psi_2' : \Psi_2''$, or (3) $\Psi_1 = \varphi_1 \mathbin{?} \Psi_1' : \Psi_1''$ and $\Psi_2 = \varphi_2 \mathbin{?} \Psi_2' : \Psi_2''$. The first two cases are symmetric and use axioms $(S1)$, $(S3)$, and $(S4)$, whereas the last case uses Lemma 5. ◀

4.3 core-wMSO

We present a complete axiomatization of a fragment of core-wMSO in which $+$ is the only allowed sum operator. Let core-wMSO$(?, +)$ be the fragment of core-wMSO given by

$$\Phi ::= \mathbf{0} \mid \textstyle\prod_x \Psi \mid \varphi \mathbin{?} \Phi_1 : \Phi_2 \mid \Phi_1 + \Phi_2,$$

where Ψ is a step-wMSO formula. Given a set of MSO formulas Γ, let $\forall x(\Gamma) = \{\forall x.\varphi \mid \varphi \in \Gamma\}$. Furthermore, for a formula Φ, let $\mathtt{var}(\Phi)$ be the set of variables used in Φ, and let $\Phi[y/x]$ be the formula resulting from replacing the variable x with the variable y. The axioms for the fragment core-wMSO$(?, +)$ are then given in Table 3. Axioms $(C1)$-$(C3)$ give standard properties of sum, whereas $(C4)$ and $(C5)$ take care of the product. Axioms $(C6)$-$(C9)$ are similar to the axioms for step-wMSO, and finally, axiom $(C10)$ shows how sum distributes over the conditional operator.

Since all of the axioms for step-wMSO are also included in the axiomatization for core-wMSO (because both include the conditional operator), we get that the theorems we derived in Proposition 4 are also derivable for core-wMSO$(?, +)$.

We first prove a lemma that shows the connection between the product operator $\prod_x$ and the first-order universal quantifier $\forall x$, and which shows that axiom $(C4)$ is sound.

▶ **Lemma 7.** $\prod_x \Psi_1 \sim_\Gamma \prod_x \Psi_2$ *if and only if* $\Psi_1 \sim_{\forall x(\Gamma)} \Psi_2$.

A key part of the proof of completeness is to put formulas into the following notion of normal form, where occurrences of the conditional operator are grouped together and all come before any sum or product is applied.

$(C1)$:	$\Gamma \vdash \Phi + \mathbf{0} \approx \Phi$	
$(C2)$:	$\Gamma \vdash \Phi_1 + \Phi_2 \approx \Phi_2 + \Phi_1$	
$(C3)$:	$\Gamma \vdash (\Phi_1 + \Phi_2) + \Phi_3 \approx \Phi_1 + (\Phi_2 + \Phi_3)$	
$(C4)$:	$\Gamma \vdash \prod_x \Psi_1 \approx \prod_x \Psi_2$	if $\forall x(\Gamma) \vdash \Psi_1 \approx \Psi_2$
$(C5)$:	$\Gamma \vdash \prod_x \Psi \approx \prod_y \Psi[y/x]$	if $y \notin \mathtt{var}(\Psi)$
$(C6)$:	$\Gamma \vdash \Phi_1 \approx \Phi_2$ implies $\Gamma \cup \{\varphi\} \vdash \Phi_1 \approx \Phi_2$	for any MSO formula φ
$(C7)$:	$\Gamma \vdash \neg\varphi \,?\, \Phi_1 : \Phi_2 \approx \varphi \,?\, \Phi_2 : \Phi_1$	
$(C8)$:	$\Gamma \vdash \varphi \,?\, \Phi_1 : \Phi_2 \approx \Phi_1$	if $\Gamma \vdash \varphi$
$(C9)$:	if $\Gamma \cup \{\varphi\} \vdash \Phi_1 \approx \Phi$ and $\Gamma \cup \{\neg\varphi\} \vdash \Phi_2 \approx \Phi$, then $\Gamma \vdash \varphi \,?\, \Phi_1 : \Phi_2 \approx \Phi$	
$(C10)$:	$\Gamma \vdash (\varphi \,?\, \Phi' : \Phi'') + \Phi \approx \varphi \,?\, \Phi' + \Phi : \Phi'' + \Phi$	

Table 3 Axioms for core-wMSO$(?, +)$.

▶ **Definition 8.** *A core-wMSO$(?, +)$ formula Φ is in* normal form *if Φ is generated by the following grammar:* $N ::= \varphi \,?\, N_1 : N_2 \mid M \mid \mathbf{0}$ *and* $M ::= \prod_x \Psi \mid M_1 + M_2$.

Every core-wMSO$(?, +)$ has an equivalent normal form, which will allow us to only reason about formulas in normal form in the proof.

▶ **Lemma 9.** *For each Γ and core-wMSO$(?, +)$ formula Φ, there exists a formula Φ' in normal form such that $\Gamma \vdash \Phi \approx \Phi'$.*

The case where both of the formulas considered in the completeness proof are conditionals is handled by the following analogue of Lemma 5.

▶ **Lemma 10.** *If $\varphi_1 \,?\, \Phi_1' : \Phi_1'' \sim_\Gamma \varphi_2 \,?\, \Phi_2' : \Phi_2''$, then*

$$\Phi_1' \sim_{\Gamma\cup\{\varphi_1,\varphi_2\}} \Phi_2', \quad \Phi_1' \sim_{\Gamma\cup\{\varphi_1,\neg\varphi_2\}} \Phi_2'', \quad \Phi_1'' \sim_{\Gamma\cup\{\neg\varphi_1,\varphi_2\}} \Phi_2', \textit{ and } \Phi_1'' \sim_{\Gamma\cup\{\neg\varphi_1,\neg\varphi_2\}} \Phi_2''.$$

Notice that for formulas in normal form, if it is not the case that $\Phi = \varphi \,?\, \Phi_1 : \Phi_2$, then Φ can not contain any conditional statements at all, and hence Φ must be of the form $\Phi = \sum_{i=1}^k \prod_x \Phi_i$ for some k (axioms (C2) and (C3) allow us to use this finite sum notation). The following lemma shows that for formulas of this form, it is enough to consider each of the summands pairwise.

▶ **Lemma 11.** *Let Γ be finite. Assume $\Phi_1 = \sum_{i=1}^k \prod_x \Psi_i$ and $\Phi_2 = \sum_{j=1}^k \prod_x \Psi_j'$ with $\Phi_1 \sim_\Gamma \Phi_2$, and assume that for all i and j $\prod_x \Psi_i \sim_\Gamma \prod_x \Psi_j'$ implies $\Gamma \vdash \prod_x \Psi_i \approx \prod_x \Psi_j'$. Then $\Gamma \vdash \Phi_1 \approx \Phi_2$.*

We can now prove completeness for formulas in normal form, and by Lemma 9, this extends to all formulas.

▶ **Lemma 12.** *If Φ_1 and Φ_2 are in normal form and Γ is finite, then $\Phi_1 \sim_\Gamma \Phi_2$ implies $\Gamma \vdash \Phi_1 \approx \Phi_2$.*

Proof sketch. The proof proceeds similarly to the proof of Theorem 6 by induction on the maximum syntactic depth of Φ_1 and Φ_2. Axiom $(C7)$ is used to handle the case where Γ is inconsistent. In the base case, if both formulas are $\mathbf{0}$, the result follows from reflexivity. Otherwise, $\Phi_1 = \prod_x \Psi_1$ and $\Phi_2 = \prod_x \Psi_2$, and by Lemma 7 and Theorem 6, this implies $\forall x(\Gamma) \vdash \Psi_1 \approx \Psi_2$, so $\Gamma \vdash \Phi_1 \approx \Phi_2$ by axiom $(C4)$ (here we use $(C5)$ implicitly to ensure that Φ_1 and Φ_2 agree on the variable x).

For the inductive step, we proceed by a case analysis of the structure of the formulas. The cases where $\Phi_1 = \Phi_1' + \Phi_1''$ and either $\Phi_2 = \mathbf{0}$ or $\Phi_2 = \prod_x \Psi$ can not happen due to the assumptions of the lemma. If $\Phi_1 = \Phi_1' + \Phi_1''$ and $\Phi_2 = \varphi_2 \,?\, \Phi_2' : \Phi_2''$, the result follows by an inductive argument on the number of conditional operators appearing in Φ_2 and using Lemma 11 and axiom $(C9)$. If $\Phi_1 = \Phi_1' + \Phi_1''$ and $\Phi_2 = \Phi_2' + \Phi_2''$, then the result follows directly by Lemma 11, which uses axioms $(C2)$, $(C3)$, $(C6)$, $(C8)$, and $(C9)$.

The cases where $\Phi_1 = \varphi_1 \,?\, \Phi_1' : \Phi_1''$ and either $\Phi_2 = \mathbf{0}$ or $\Phi_2 = \prod_x \Psi$ are taken care of by axiom $(C9)$. The case where both formulas are conditional formulas is handled by Lemma 10, and the remaining cases are symmetric version of cases that have already been considered. ◀

▶ **Theorem 13** (Completeness). *For finite Γ we have $\Phi_1 \sim_\Gamma \Phi_2$ if and only if $\Gamma \vdash \Phi_1 \approx \Phi_2$.*

Proof. Assume $\Phi_1 \sim_\Gamma \Phi_2$. By Lemma 9, which uses axioms $(C1)$ and $(C10)$, there exist formulas Φ_1' and Φ_2', both in normal form, such that $\Gamma \vdash \Phi_1 \approx \Phi_1'$ and $\Gamma \vdash \Phi_2 \approx \Phi_2'$. By soundness, this implies $\Phi_1 \sim_\Gamma \Phi_1'$ and $\Phi_2 \sim_\Gamma \Phi_2'$, so $\Phi_1' \sim_\Gamma \Phi_2'$. Since these are in normal form, Lemma 12 gives $\Gamma \vdash \Phi_1' \approx \Phi_2'$, and by symmetry and transitivity, this implies $\Gamma \vdash \Phi_1 \approx \Phi_2$. ◀

5 Decision problems

The three usual decision problems that one considers for a logical language are model checking, satisfiability, and validity. The model checking problem asks whether a given model satisfies a given formula, the satisfiability problem asks whether for a given formula there exists a model which satisfies the formula, and the validity problem asks whether a given formula is satisfied by all models. For FO, MSO, and many classical Boolean logics, the satisfiability and validity problems are equivalent, since the satisfiability of a formula φ is equivalent to the non-validity of its negation $\neg\varphi$.

In this section, we discuss how to extend these fundamental notions to our setting of a real-valued logic. We assume the set R of weights has decidable equality, i.e. it is decidable (and with reasonable efficiency, when discussing complexity issues) whether $r_1 = r_2$ for two weights $r_1, r_2 \in R$. First, we observe that we can encode every MSO formula as an equation.

▶ **Lemma 14.** *Assume two distinct values, $0, 1 \in R$, and let $\varphi \in$ MSO. Then, for every (w, σ), the following are equivalent: **(1)** $(w, \sigma) \models \varphi$, **(2)** $[\![\varphi \,?\, 0 : 0]\!]\,(w, \sigma) = [\![\varphi \,?\, 0 : 1]\!]\,(w, \sigma)$, and **(3)** $[\![\varphi \,?\, \Pi_x 0 : \Pi_x 0]\!]\,(w, \sigma) = [\![\varphi \,?\, \Pi_x 0 : \Pi_x 1]\!]\,(w, \sigma)$.*

In a certain sense, step-wMSO can also be described using MSO.

▶ **Definition 15.** *For $\Psi \in$ step-wMSO and $r \in \mathcal{R}$, we define $\varphi(\Psi, r)$ recursively: $\varphi(r, r) = \top$ and $\varphi(r', r) = \neg\top$, when $r \neq r'$; and $\varphi(\varphi' \,?\, \Psi_1 : \Psi_2, r) = (\varphi' \wedge \varphi(\Psi_1, r)) \vee (\neg\varphi' \wedge \varphi(\Psi_2, r))$.*

▶ **Lemma 16.** *$(w, \sigma) \in [\![\varphi(\Psi, r)]\!]$ iff $[\![\Psi]\!]\,(w, \sigma) = r$.*

We consider weighted model checking, which has two versions. We recall that for MSO and FO, model checking is PSPACE-complete [19, 21].

▶ **Definition 17** (The evaluation problem). *Given (w, σ) and a formula Φ, compute $[\![\Phi]\!]\,(w, \sigma)$.*

▶ **Definition 18** (Weighted model checking problem). *Given (w, σ), a formula Φ, and r, do we have $[\![\Phi]\!]\,(w, \sigma) = r$?*

To evaluate a step-wMSO or core-wMSO formula on (w, σ), one can use the recursive procedure that is yielded by the semantics of step-wMSO and core-wMSO, using a model checking algorithm for MSO to check which branch to take at each conditional. It is not hard to see that for step-wMSO, this can be done using polynomial space, as that fragment only uses conditionals on MSO formulas and values. Then, the weighted model checking problem is decidable and PSPACE-complete for step-wMSO, using Lemmata 14 and 16.

As for weighted model checking for core-wMSO, one has to determine what r can be, and how it is described. Naturally, for the abstract semantics that we use, it does not make much sense to assume that $r \in R$, because then, due to the syntax and semantics of core-wMSO, the value that Φ returns can never be a member of R and the problem becomes trivial. If r is a multiset of tuples from R, or a set of such values, possibly described by a formula, the weighted model checking problem can still be reduced to the evaluation problem.

Next we consider several variations of the satisfiability problem in the weighted setting.

▶ **Definition 19** (Weighted satisfiability). *Given (w, σ) and Φ, does there exist r such that $[\![\Phi]\!](w, \sigma) = r$?*

Since $[\![\cdot]\!]$ is a total function, the answer is always yes, so this variation is not interesting.

▶ **Definition 20** (r-satisfiability). *Given Φ and r, is there (w, σ) such that $[\![\Phi]\!](w, \sigma) = r$?*

For step-wMSO formulas, this problem has the same complexity as MSO satisfiability over finite strings, using Lemmata 16 and 14. Therefore, the problem is decidable, but with a nonelementary complexity [18]. For core-wMSO formulas Φ, similarly to the case of weighted model checking, the problem is not as interesting for a single value $r \in R$, and therefore the following variation is more interesting,

▶ **Definition 21** (Equational satisfiability). *Given Φ_1 and Φ_2, does there exist (w, σ) such that $[\![\Phi_1]\!](w, \sigma) = [\![\Phi_2]\!](w, \sigma)$?*

For step-wMSO formulas, this problem is decidable in the same way as r-satisfiability, by reducing to the satisfiability problem of MSO: there exist (w, σ) such that $[\![\Phi_1]\!](w, \sigma) = [\![\Phi_2]\!](w, \sigma)$ if and only if

$$\bigvee_{\substack{r \text{ appears} \\ \text{in } \Psi_1 \text{ and } \Psi_2}} \varphi(\Psi_1, r) \wedge \varphi(\Psi_2, r)$$

is satisfiable. For core-wMSO, and even core-wFO formulas, we show that this problem is undecidable in Section 6. Finally, we consider variations of validity in the weighted setting.

▶ **Definition 22** (Weighted validity). *Given Φ, does there exist an r such that $[\![\Phi]\!](w, \sigma) = r$ for all (w, σ)?*

This problem is decidable for Ψ and core-wMSO$(?, +)$. As we gave a recursive and complete axiomatization of the equational theory of these fragments, the problem is recursively enumerable (RE). But the logic also has a recursive set of models and a decidable evaluation problem. Therefore, this version of validity is also coRE, and therefore decidable.

Similarly to the various versions of weighted satisfiability and model checking above, the problem becomes trivial for core-wMSO, for a single value $r \in R$, and therefore the following variation suits better the full weighted logic.

▶ **Definition 23** (Equational validity). *Given Φ_1 and Φ_2, do we have $[\![\Phi_1]\!](w, \sigma) = [\![\Phi_2]\!](w, \sigma)$ for all (w, σ)?*

For step-wMSO formulas, this problem is decidable, via the same argument as r-validity. For core-wMSO formulas, we discuss in Section 6 why it is likely that this variation is undecidable. Note that the undecidability of this problem would imply that there is no recursive sound and complete axiomatization of the full core-wMSO logic.

6 Equational Satisfiability is Undecidable

In this section, we prove that equational satisfiability for the full core-wMSO is undecidable. Furthermore, this is the case, even for core-wFO. We first observe that, if we assume an unbounded set of values, the language of equations is closed under conjunction with respect to satisfiablity, in the sense of Lemma 24.

▶ **Lemma 24.** *Let $\Phi_1, \Phi_2, \Phi_1', \Phi_2' \in$ core-wMSO be such that Φ_1, Φ_2 use values that are distinct from the ones that Φ_1', Φ_2' use. For every w, σ, $[\![\Phi_1]\!](w,\sigma) = [\![\Phi_2]\!](w,\sigma)$ and $[\![\Phi_1']\!](w,\sigma) = [\![\Phi_2']\!](w,\sigma)$, if and only if $[\![\Phi_1 + \Phi_1']\!](w,\sigma) = [\![\Phi_2 + \Phi_2']\!](w,\sigma)$.*

Fix a pair (w,σ). We use a series of formulas and equations to express that a pair (w,σ) encodes the computation of a Turing Machine that halts. Therefore, the question of whether there is such a pair that satisfies the resulting set of equations is undecidable. Let $T = (Q, \Sigma, \delta, q_0, H)$ be a Turing Machine, where Q is a finite set of states, Σ is the set of symbols that the machine uses, $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, R\}$ is the machine's transition function, q_0 is the starting state, and H is the halting state of T. We give the construction of the core-wMSO formula equations. The full construction for the case of core-wFO is given in the appendix.

Let $\triangleleft$ be a special symbol not in Σ. A configuration of T is represented by a string of the form $s_1 q s_2 \triangleleft$, where q is the current state for the configuration, $s_1 s_2$ is the string of symbols in the tape of the machine, and the head is located at the first symbol of s_2; $\triangleleft$ marks the end of the configuration. Let $x_0 \in \Sigma^*$ be an input of T.

We use every $s \in Q \cup \Sigma \cup \{\triangleleft\}$ as a predicate, so that $s(x)$ is true if and only if the symbol s is in position x. We want to describe that (w,σ) encodes a halting run of T on x_0. In other words, we must ensure that (w,σ) is a sequence $c_0 \cdots c_k$ of configurations of T, such that c_0 is $q_0 x_0 \triangleleft$ and c_k is $s_1 H s_2 \triangleleft$, where $s_1, s_2 \in \Sigma^*$.

We must therefore ensure that the following conditions hold:

1. (w,σ) is of the form $c_0 c_1 \cdots c_k$, where each c_i has exactly one $\triangleleft$, at the end;
2. each c_i is of the form $s_1 q s_2 \triangleleft$, where $q \in Q$, $s_1 s_2 \in \Sigma^*$, and $s_2 \neq \varepsilon$;
3. $c_0 = q_0 x_0 \triangleleft$;
4. $c_k = s_1 H s_2 \triangleleft$ for some s_1, s_2; and
5. for every $0 \leq i < k$, c_{i+1} results from c_i by applying the transition function δ. This condition can be further refined into the following subconditions. For every $0 \leq i < k$, if $c_i = x_1\ x_2 \cdots\ x_r\ q_i\ y_1\ y_2 \cdots\ y_{r'} \triangleleft$, then:
   a. if $\delta(q_i, y_1) = (q, x, L)$ and $r > 0$, then $c_{i+1} = x_1\ x_2 \cdots\ x_{r-1}\ q\ x_r\ x\ y_2 \cdots\ y_{r'} \triangleleft$,
   b. if $\delta(q_i, y_1) = (q, x, L)$ and $r = 0$, then $c_{i+1} = q\ x\ y_2 \cdots\ y_{r'} \triangleleft$,
   c. if $\delta(q_i, y_1) = (q, x, R)$ and $r' > 1$, then $c_{i+1} = x_1\ x_2 \cdots\ x_r\ x\ q\ y_2 \cdots\ y_{r'} \triangleleft$, and
   d. if $\delta(q_i, y_1) = (q, x, R)$ and $r' = 1$, then $c_{i+1} = x_1\ x_2 \cdots\ x_r\ x\ q\ \_\triangleleft$, where $\_ \in \Sigma$ is the symbol used by T for a blank space.

We now explain how to represent each of the conditions above with a formula or equation. We use the following macros, where $0, 1 \in R$ are two distinct weights:

$$\mathtt{nxt}(x,y) \stackrel{\mathsf{def}}{=} (\neg(y \le x)) \wedge \forall z.\ (z \le x \vee y \le z) \qquad \mathtt{last}(x) \stackrel{\mathsf{def}}{=} \forall y.\ y \le x$$

$$\mathtt{first}(x) \stackrel{\mathsf{def}}{=} \forall y.\ y \ge x \quad \text{and} \quad \mathtt{first\text{-}cf}(x) \stackrel{\mathsf{def}}{=} \mathtt{first}(x) \vee \exists y.\ \lhd(y) \wedge \mathtt{nxt}(y,x)$$

$$v_1(x) \stackrel{\mathsf{def}}{=} \prod_y (x = y)\ ?\ 1 : 0, \quad v_s^x \stackrel{\mathsf{def}}{=} \prod_y (x = y)\ ?\ s : 0, \quad \text{and} \quad v_0 \stackrel{\mathsf{def}}{=} \mathbf{0}$$

$$\mathsf{first\text{-}cf\text{-}x}(x,y) \stackrel{\mathsf{def}}{=} \mathtt{first\text{-}cf}(y) \wedge y \le x \wedge \forall z. \neg(\lhd(z) \wedge y \le z \le x)$$

$$\mathtt{ps}_v(x) \stackrel{\mathsf{def}}{=} \sum_X \exists y.\ \mathsf{first\text{-}cf\text{-}x}(x,y) \wedge \forall z.(\neg z \in X) \vee (y \le z \le x)\ ?\ v : v_0$$

Intuitively, formula $\mathtt{ps}_v(x)$ counts 2^i, where i is the position of x in its configuration. We note that for each symbol s that appears in the set S of positions in a configuration, S is uniquely identified by $\sum_{i \in S} 2^i$. Furthermore, for each configuration, $\mathtt{ps}_v(x)$ constructs a map from each such s (represented by the returned value v) to $\sum_{i \in S} 2^i$. Therefore, the way that we will use $\mathtt{ps}_v(x)$ (see how we deal with condition 5, below) gives a complete description of each configuration. We will use v_0 as the default (negative) value in conditionals, and as such $\varphi\ ?\ v$ is used as shorthand for $\varphi\ ?\ v : v_0$. Furthermore, we assume that $:$ binds to the nearest ?, and therefore, $\varphi_1\ ?\ \varphi_2\ ?\ \Phi_1 : \Phi_2$ means $\varphi_1\ ?\ \varphi_2\ ?\ \Phi_1 : \Phi_2 : v_0$, which can be uniquely parsed as $\varphi_1\ ?\ (\varphi_2\ ?\ \Phi_1 : \Phi_2) : v_0$.

We now proceed to describe, for each of the conditions 1-6, a number of equations that ensure that this condition holds. By an equation, we mean something of the form $\Phi = \Phi'$, where Φ and Φ' are core-wMSO formulas. Notice that by Lemma 14, any MSO formula can be turned into an equation (as long as we have at least two distinct weights), so for some conditions we give an MSO formula rather than an equation.

A number of equations $\Phi_i = \Phi_i'$ ensures that the condition holds in the sense that for any (w, σ), $[\![\Phi_i]\!]\,(w,\sigma) = [\![\Phi_i']\!]\,(w,\sigma)$ for each i if and only if (w, σ) satisfies the condition. By Lemma 24, once we have a number of equations $\Phi_i = \Phi_i'$ that together ensure that all conditions are satisfied, the equation $\sum_i \Phi_i = \sum_i \Phi_i'$ ensures that all conditions are satisfied, so that (w, σ) satisfies the conditions if and only if $[\![\sum_i \Phi_i]\!]\,(w,\sigma) = [\![\sum_i \Phi_i']\!]\,(w,\sigma)$. We omit most conditions, as it is not hard to express then in FO, and only demonstrate how to treat case a of condition 5. The other cases are analogous.

Fix a transition $(q, s, q's', L) \in \delta$ and $d \in \Sigma$. We use the following shorthand.

$$\mathtt{tr}(x,y,z) \stackrel{\mathsf{def}}{=} q(y) \wedge y \le x \wedge \forall y'.\ \neg(\lhd(y') \wedge y \le y' \le x) \wedge s(z) \wedge \mathtt{nxt}(y,z) \quad \text{and}$$

$$\mathtt{tr'}(x,y,z) \stackrel{\mathsf{def}}{=} q'(y) \wedge y \le x \wedge \forall y'.\ \neg(\lhd(y') \wedge y \le y' \le x) \wedge s'(z) \wedge \mathtt{nxt}(y,z).$$

Let $s_1, s_2, \ldots, s_m$ be a permutation of Σ. We use the following equation:

$$\begin{aligned}
&\sum_x \lhd(x) \wedge \exists y.(\lhd(y) \wedge x{<}y) \wedge \exists y,z.\mathtt{tr}(x,y,z)\ ?\ \sum_y y{\le}x \wedge \forall z.(x{\le}z \vee z{<}y \vee \neg\lhd(x))? \\
&\qquad q(y)\ ?\ \mathtt{ps}_{v_q^x}(y) : s_1(y)\ ?\ \mathtt{ps}_{v_{s_1}^x}(y) : \cdots : s_m(y)\ ?\ \mathtt{ps}_{v_{s_m}^x}(y) \qquad = \\
&\sum_x \lhd(x) \wedge \exists y.(\lhd(y) \wedge x{<}y) \wedge \exists y,z.\mathtt{tr}(x,y,z)\ ?\ \sum_y x{\le}y \wedge \forall z.(z{\le}x \vee y{<}z \vee \neg\lhd(x))? \\
&\quad q'(y) \wedge \exists z.\mathtt{nxt}(y,z) \wedge s_1(z)\ ?\ \mathtt{ps}_{v_{s_1}^x}(y) : \cdots q'(y) \wedge \exists z.\mathtt{nxt}(y,z) \wedge s_m(z)\ ?\ \mathtt{ps}_{v_{s_m}^x}(y) : \\
&\qquad \exists z.q'(z) \wedge \mathtt{nxt}(z,y)\ ?\ \mathtt{ps}_{v_q^x}(y) : \exists z,z'.q'(z) \wedge \mathtt{nxt}(z,z') \wedge \mathtt{nxt}(z',y)\ ?\ \mathtt{ps}_{v_s^x}(y) : \\
&\qquad\quad s_1(y)\ ?\ \mathtt{ps}_{v_{s_1}^x}(y) : \cdots : s_m(y)\ ?\ \mathtt{ps}_{v_{s_m}^x}(y)
\end{aligned}$$

The rightmost part of the equation ensures that if the effects of the transition are reversed, then all symbols are in the same place as in the previous configuration. We can then make sure that the state has changed to q' and the symbol to s' with the following formula:

$$\forall x, y.\ \neg(\triangleleft(x) \wedge \triangleleft(y) \wedge \neg y \leq x \wedge \exists x_q, x_s.\ \texttt{tr}(x, x_q, x_s)) \vee \exists y_q, y_s.\ \texttt{tr'}(y, y_q, y_s).$$

▶ **Theorem 25.** *If the set of weights has at least two distinct weights, the equational satisfiability problem for* core-wMSO *and* core-wFO *is undecidable.*

Proof. For the case of core-wMSO, we use a reduction from the Halting Problem, as it is described above. It is not hard to see why conditions 1 to 5 suffice for the correctness of the reduction. The full construction for the case of core-wFO can be found in the appendix. ◀

Completeness, Decidability of Equational Validity, and Semantics Theorem 25 does not inform us whether equational validity for core-wMSO and core-wFO is decidable or not. If we can express in the equational language the negation of an equation, then we can use Theorem 25 to prove the undecidability of equational validity for these weighted logics. That result would then imply that there is no recursive and complete axiomatization for the full core-wMSO (and core-wFO) — otherwise, we reach a contradiction, because equational validity would be RE and coRE at the same time. Although for the abstract semantics we do not have an undecidability proof for this problem, it is not hard to see that there are computationally reasonable concrete semantics for which we can express with an equation that two formulas return different values: consider a difference operation that returns 0 or 1, depending on whether two sets of values match or not.

▶ **Example 26.** Consider the semiring of integers $(\mathbb{Z}, +, \times, 0, 1)$ and formulas Φ_1 and Φ_2 that output positive values, and Φ_3 (or $-\Phi_2$) that outputs the values of Φ_2 with a negative sign. We also have $\chi_0(n) = 0$ if $n = 0$ and 1 otherwise. Then, $[\![\Phi_1]\!](w, \sigma) = [\![\Phi_2]\!](w, \sigma)$ if and only if $[\![\Phi_1]\!](w, \sigma) - [\![\Phi_2]\!](w, \sigma)(= [\![\Phi_1]\!](w, \sigma) + [\![\Phi_3]\!](w, \sigma)) = 0$ and $[\![\Phi_1]\!](w, \sigma) \neq [\![\Phi_2]\!](w, \sigma)$ if and only if $\chi_0([\![\Phi_1]\!](w, \sigma) - [\![\Phi_2]\!](w, \sigma)) = 1$.

7 Conclusion

We have given a sound and complete axiomatization of the fragment of weighted monadic second-order logic in which generalized sum is not allowed. This leaves open the problem of finding a complete axiomatization for the full logic. However, we conjecture that no recursive axiomatization can also be complete. Furthermore, we have investigated weighted versions of common decision problems for logics, specifically model checking, satisfiability, and validity. For the second layer of the logic, step-wMSO, we have shown that these problems are all decidable, although many of them have non-elementary complexity inherited from the corresponding problems for first- and second-order logic. For the third layer, core-wMSO, things are less clear. We have demonstrated some of the problems to be decidable, but we have also shown that the problem of deciding whether there exists an input that makes two given formulas return the same value is undecidable, and we conjecture that the related problem of deciding whether two formulas return the same value for all inputs is also undecidable. If the latter undecidability result holds, this will imply that there can be no recursive and complete axiomatization for the full weighted monadic second-order logic, and hence this problem is of particular interest. Other open questions of interest are to discover how different concrete semantics affect the decidability of equational satisfiability and validity, as well as to consider other useful relations of formulas.

References

1 J. Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960. doi:10.1002/malq.19600060105.

2 Manfred Droste and Paul Gastin. Weighted automata and weighted logics. *Theor. Comput. Sci.*, 380(1-2):69–86, 2007. doi:10.1016/j.tcs.2007.02.055.

3 Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98(1):21–51, 1961. URL: http://www.jstor.org/stable/1993511.

4 Ina Fichtner. Weighted picture automata and weighted logics. *Theory Comput. Syst.*, 48(1):48–78, 2011. doi:10.1007/s00224-009-9225-3.

5 Paul Gastin and Benjamin Monmege. A unifying survey on weighted logics and weighted automata - core weighted logic: minimal and versatile specification of quantitative properties. *Soft Comput.*, 22(4):1047–1065, 2018. doi:10.1007/s00500-015-1952-6.

6 Amélie Gheerbrant and Balder ten Cate. Complete axiomatizations of fragments of monadic second-order logic on finite trees. *Logical Methods in Computer Science*, 8(4), 2012. doi:10.2168/LMCS-8(4:12)2012.

7 Mikkel Hansen, Kim Guldstrand Larsen, Radu Mardare, and Mathias Ruggaard Pedersen. Reasoning about bounds in weighted transition systems. *Logical Methods in Computer Science*, 14(4), 2018. doi:10.23638/LMCS-14(4:19)2018.

8 Karel Culik II and Jarkko Kari. Image compression using weighted finite automata. *Comput. Graph.*, 17(3):305–313, 1993. doi:10.1016/0097-8493(93)90079-O.

9 Kevin Knight and Jonathan May. Applications of weighted automata in natural language processing. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, pages 571–596. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. doi:10.1007/978-3-642-01492-5_14.

10 Stephan Kreutzer and Cristian Riveros. Quantitative monadic second-order logic. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 113–122. IEEE Computer Society, 2013. doi:10.1109/LICS.2013.16.

11 Kim G. Larsen and Radu Mardare. Complete proof systems for weighted modal logic. *Theor. Comput. Sci.*, 546:164–175, 2014. doi:10.1016/j.tcs.2014.03.007.

12 Kim G. Larsen, Radu Mardare, and Bingtian Xue. Alternation-free weighted mu-calculus: Decidability and completeness. In Dan R. Ghica, editor, *The 31st Conference on the Mathematical Foundations of Programming Semantics, MFPS 2015, Nijmegen, The Netherlands, June 22-25, 2015*, volume 319 of *Electronic Notes in Theoretical Computer Science*, pages 289–313. Elsevier, 2015. doi:10.1016/j.entcs.2015.12.018.

13 Kim G. Larsen, Radu Mardare, and Bingtian Xue. Concurrent weighted logic. *J. Log. Algebraic Methods Program.*, 84(6):884–897, 2015. doi:10.1016/j.jlamp.2015.07.002.

14 Kim G. Larsen, Radu Mardare, and Bingtian Xue. On decidability of recursive weighted logics. *Soft Comput.*, 22(4):1085–1102, 2018. doi:10.1007/s00500-016-2193-z.

15 Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004. doi:https://doi.org/10.1007/978-3-662-07003-1.

16 Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted automata in text and speech processing. In *Proceedings of the 12th biannual European Conference on Artificial Intelligence (ECAI 96), Workshop on Extended finite state models of language, Budapest, Hungary*. John Wiley & Sons, Ltd., 1996.

17 George Rahonis. Weighted Muller tree automata and weighted logics. *Journal of Automata, Languages and Combinatorics*, 12(4):455–483, 2007. doi:10.25596/jalc-2007-455.

18 Klaus Reinhardt. The complexity of translating logic to finite automata. In Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors, *Automata Logics, and Infinite Games: A Guide to Current Research*, pages 231–238. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. doi:10.1007/3-540-36387-4_13.

19 Larry J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory*. PhD thesis, MIT, 1974.

20 Boris A. Trakhtenbrot. Finite automata and the logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 149:326–329, 1961.

21 Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 137–146. ACM, 1982. doi:10.1145/800070.802186.

Appendix

A Proofs for the Results of Section 4: Soundness and Completeness

Proof of Corollary 3. ($\Longrightarrow$) $\Gamma \models \varphi$ means that for all $(w,\sigma) \in [\![\Gamma]\!] = [\![\bigwedge \Gamma]\!]$, $(w,\sigma) \models \varphi$. Hence for any (w,σ), $(w,\sigma) \in [\![\bigwedge \Gamma]\!]$ implies $(w,\sigma) \in [\![\varphi]\!]$, so $\models \bigwedge \Gamma \to \varphi$. By Theorem 2 this implies $\vdash \bigwedge \Gamma \to \varphi$, which by definition means $\Gamma \vdash \varphi$.

($\Longleftarrow$) $\Gamma \vdash \varphi$ means that there is a finite subset $\Gamma' \subseteq \Gamma$ such that $\vdash \bigwedge \Gamma' \to \varphi$. By Theorem 2 this implies $\models \bigwedge \Gamma' \to \varphi$, so $\Gamma' \models \varphi$. Since $\Gamma' \subseteq \Gamma$, we get $\Gamma \models \varphi$. ◀

Proof of Proposition 4. **1.** Let Ψ_1 and Ψ_2 be arbitrary step-wMSO formulas and assume that Γ is inconsistent. Then $\Gamma \vdash \varphi$ and $\Gamma \vdash \neg\varphi$. Then axiom $(S3)$ gives

$$\Gamma \vdash \varphi \,?\, \Psi_1 : \Psi_2 \approx \Psi_1 \quad \text{and} \quad \Gamma \vdash \neg\varphi \,?\, \Psi_2 : \Psi_1 \approx \Psi_2.$$

Since $\Gamma \vdash \varphi \,?\, \Psi_1 : \Psi_2 \approx \neg\varphi \,?\, \Psi_2 : \Psi_1$ by axiom $(S2)$, this implies $\Gamma \vdash \Psi_1 \approx \Psi_2$.

2. We have assumed $\Gamma \cup \{\varphi\} \vdash \Psi_1 \approx \Psi_2$, and we get $\Gamma \cup \{\neg\varphi\} \vdash \Psi_2 \approx \Psi_2$ by reflexivity, so $(S4)$ gives $\Gamma \vdash \varphi \,?\, \Psi_1 : \Psi_2 \approx \Psi_2$. Since we have assumed $\Gamma \vdash \varphi$, $(S3)$ gives $\Gamma \vdash \varphi \,?\, \Psi_1 : \Psi_2 \approx \Psi_1$. Hence $\Gamma \vdash \Psi_1 \approx \Psi_2$ by symmetry and transitivity.

3. By reflexivity, we have $\Gamma \vdash \Psi \approx \Psi$, so $(S1)$ gives both $\Gamma \cup \{\varphi\} \vdash \Psi \approx \Psi$ and $\Gamma \cup \{\neg\varphi\} \vdash \Psi \approx \Psi$, so using $(S4)$ we conclude $\Gamma \vdash \varphi \,?\, \Psi : \Psi \approx \Psi$.

4. Using $(S4)$, $\Gamma \cup \{\varphi_1, \varphi_2\} \vdash \Psi_1 \approx \Psi_1'$ and $\Gamma \cup \{\varphi_1, \neg\varphi_2\} \vdash \Psi_1 \approx \Psi_2'$ gives $\Gamma \cup \{\varphi_1\} \vdash \varphi_2 ? \Psi_1' : \Psi_2' \approx \Psi_1$. Likewise, using the other two assumptions, we get $\Gamma \cup \{\neg\varphi_1\} \vdash \varphi_2 ? \Psi_1' : \Psi_2' \approx \Psi_2$, and a final application of $(S4)$ then gives $\Gamma \vdash \varphi_1 \,?\, \Psi_1 : \Psi_2 \approx \varphi_2 \,?\, \Psi_1' : \Psi_2'$.

5. Assume that $\Gamma \vdash \varphi_1 \leftrightarrow \varphi_2$. Then

$$\begin{aligned}
\Gamma \cup \{\varphi_1, \varphi_2\} &\vdash \Psi_1 \approx \Psi_1 \text{ by reflexivity,} \\
\Gamma \cup \{\varphi_1, \neg\varphi_2\} &\vdash \Psi_1 \approx \Psi_2 \text{ by inconsistency,} \\
\Gamma \cup \{\neg\varphi_1, \varphi_2\} &\vdash \Psi_2 \approx \Psi_1 \text{ by inconsistency, and} \\
\Gamma \cup \{\neg\varphi_1, \neg\varphi_2\} &\vdash \Psi_2 \approx \Psi_2 \text{ by reflexivity.}
\end{aligned}$$

Hence the fourth item of this proposition gives $\Gamma \vdash \varphi_1 \,?\, \Psi_1 : \Psi_2 \approx \varphi_2 \,?\, \Psi_1 : \Psi_2$.

6. Assume that $\Gamma \vdash \neg\varphi$. By axiom $(S2)$ we get $\Gamma \vdash \varphi \,?\, \Psi_1 : \Psi_2 \approx \neg\varphi \,?\, \Psi_2 : \Psi_1$, and axiom $(S3)$ gives $\Gamma \vdash \neg\varphi \,?\, \Psi_2 : \Psi_1 \approx \Psi_2$, so $\Gamma \vdash \varphi \,?\, \Psi_1 : \Psi_2 \approx \Psi_2$.

7. This is simply an instantiation of the fourth item of this proposition where $\varphi_1 = \varphi_2 = \varphi$, and the other two premises are guaranteed to hold because $\{\varphi, \neg\varphi\}$ is inconsistent.

8. Assume that $\Gamma \cup \{\varphi\} \vdash \Psi_1 \approx \Psi_2$ and $\Gamma \cup \{\neg\varphi\} \vdash \Psi_1 \approx \Psi_2$. Then axiom (S4) gives $\Gamma \vdash \varphi \,?\, \Psi_1 : \Psi_1 \approx \Psi_2$, and the third item of this proposition gives $\Gamma \vdash \varphi \,?\, \Psi_1 : \Psi_1 \approx \Psi_1$, so $\Gamma \vdash \Psi_1 \approx \Psi_2$.

9. Since $\Gamma \cup \{\varphi\} \vdash \varphi$, we get $\Gamma \cup \{\varphi\} \vdash \varphi \,?\, \Psi_1 : \Psi_2 \approx \Psi_1$ by $(S3)$. ◀

Proof of Lemma 5. We show why the first equivalence is true; the remaining cases are similar. Let $\Psi_1 = \varphi_1 \,?\, \Psi_1' : \Psi_1''$ and $\Phi_2 = \varphi_2 \,?\, \Psi_2' : \Psi_2''$. If $(w,\sigma) \in [\![\Gamma \cup \{\varphi_1, \varphi_2\}]\!]$, then also $(w,\sigma) \in [\![\Gamma]\!]$, so

$$[\![\Psi_1']\!]\,(w,\sigma) = [\![\Psi_1]\!]\,(w,\sigma) = [\![\Psi_2]\!]\,(w,\sigma) = [\![\Psi_2']\!]\,(w,\sigma). \qquad \blacktriangleleft$$

Proof of Theorem 6. $(\implies)$: Note that if Γ is inconsistent, then immediately $\Gamma \vdash \Psi_1 \approx \Psi_2$ by Proposition 4(1). In the rest of the proof we may therefore assume that Γ is consistent.

The proof now proceeds by induction on the maximum depth of Ψ_1 and Ψ_2, defined as follows.

$$\mathtt{depth}(\Psi) = \begin{cases} 0 & \text{if } \Psi = r \\ 1 + \max\{\mathtt{depth}(\Psi'), \mathtt{depth}(\Psi'')\} & \text{if } \Psi = \varphi \,?\, \Psi' : \Psi'' \end{cases}$$

Case $\max\{\mathtt{depth}(\Psi_1), \mathtt{depth}(\Psi_2)\} = 0$: In this case, $\Psi_1 = r_1$ and $\Psi_2 = r_2$ for some $r_1, r_2 \in R$. Since $r_1 = [\![\Psi_1]\!]\,(w,\sigma) = [\![\Psi_2]\!]\,(w,\sigma) = r_2$ by assumption, we get $\Gamma \vdash \Psi_1 \approx \Psi_2$ by reflexivity.

Case $\max\{\mathtt{depth}(\Psi_1), \mathtt{depth}(\Psi_2)\} > 0$: We have three subcases to consider: (1) $\Psi_1 = \varphi \,?\, \Psi_1' : \Psi_1''$ and $\Psi_2 = r_2$, (2) $\Psi_1 = r_1$ and $\Psi_2 = \varphi \,?\, \Psi_2' : \Psi_2''$, or (3) $\Psi_1 = \varphi_1 \,?\, \Psi_1' : \Psi_1''$ and $\Psi_2 = \varphi_2 \,?\, \Psi_2' : \Psi_2''$.

(1) The only ways for $[\![\Psi_1]\!]\,(w,\sigma) = r_1$ to hold for every $(w,\sigma) \in [\![\Gamma]\!]$ are
 - **(a)** $[\![\Psi_1']\!]\,(w,\sigma) = [\![\Psi_1'']\!]\,(w,\sigma) = r_2$ for every $(w,\sigma) \in [\![\Gamma]\!]$,
 - **(b)** $\Gamma \vdash \varphi$ and $[\![\Psi_1']\!]\,(w,\sigma) = r_2$ for all $(w,\sigma) \in [\![\Gamma]\!]$, or
 - **(c)** $\Gamma \vdash \neg\varphi$ and $[\![\Psi_2']\!]\,(w,\sigma) = r_2$ for all $(w,\sigma) \in [\![\Gamma]\!]$.

 We now consider each of these cases in turn.
 - **(a)** By induction hypothesis, we get $\Gamma \vdash \Psi_1' \approx r_2$ and $\Gamma \vdash \Psi_1'' \approx r_2$. By axiom $(S1)$, we get $\Gamma \cup \{\varphi\} \vdash \Psi_1' \approx r_2$ and $\Gamma \cup \{\neg\varphi\} \vdash \Psi_1'' \approx r_2$. Axiom $(S4)$ then gives $\Gamma \vdash \varphi ? \Psi_1' : \Psi_1'' \approx r_2$.
 - **(b)** The induction hypothesis gives $\Gamma \vdash \Psi_1' \approx r_2$, and axiom $(S3)$ gives $\Gamma \vdash \varphi \,?\, \Psi_1' : \Psi_1'' \approx \Psi_1'$, so we conclude $\Gamma \vdash \Psi_1 \approx r_2$.
 - **(c)** The induction hypothesis gives $\Gamma \vdash \Psi_1'' \approx r_2$, and Proposition 4(6) gives $\Gamma \vdash \varphi \,?\, \Psi_1' : \Psi_1'' \approx \Psi_1''$,s so $\Gamma \vdash \Psi_1 \approx r_2$. Putting this together, we conclude that $\Gamma \vdash \Psi_1 \approx r_2$.

(2) This case is symmetric to case (1).

(3) By Lemma 5, we get

$$\begin{array}{ll} \Psi_1' \sim_{\Gamma \cup \{\varphi_1, \varphi_2\}} \Psi_2', & \Psi_1' \sim_{\Gamma \cup \{\varphi_1, \neg\varphi_2\}} \Psi_2'', \\ \Psi_1'' \sim_{\Gamma \cup \{\neg\varphi_1, \varphi_2\}} \Psi_2', \text{ and } & \Psi_1'' \sim_{\Gamma \cup \{\neg\varphi_1, \neg\varphi_2\}} \Psi_2''. \end{array}$$

The induction hypothesis then implies that

$$\begin{array}{ll} \Gamma \cup \{\varphi_1, \varphi_2\} \vdash \Psi_1' \approx \Psi_2', & \Gamma \cup \{\varphi_1, \neg\varphi_2\} \vdash \Psi_1' \approx \Psi_2'', \\ \Gamma \cup \{\neg\varphi_1, \varphi_2\} \vdash \Psi_1'' \approx \Psi_2', \text{ and } & \Gamma \cup \{\neg\varphi_1, \neg\varphi_2\} \vdash \Psi_1'' \approx \Psi_2''. \end{array}$$

This means that all the premises for Proposition 4(3) are met, so we conclude that $\Gamma \vdash \Psi_1 \approx \Psi_2$.

$(\impliedby)$: We show the soundness of each axiom in turn.

$(S1)$: Assume that $\Psi_1 \sim_\Gamma \Psi_2$. Since $[\![\Gamma \cup \{\varphi\}]\!] = [\![\Gamma]\!] \cap [\![\varphi]\!]$, for any $(w,\sigma) \in [\![\Gamma \cup \{\varphi\}]\!]$ we have $(w,\sigma) \in [\![\Gamma]\!]$, and hence $[\![\Psi_1]\!]\,(w,\sigma) = [\![\Psi_2]\!]\,(w,\sigma)$ by assumption. We conclude that $\Psi_1 \sim_{\Gamma \cup \{\varphi\}} \Psi_2$.

$(S2)$: $\llbracket \varphi \mathbin{?} \Psi_1 : \Psi_2 \rrbracket (w,\sigma) = \llbracket \Psi_1 \rrbracket (w,\sigma)$ if and only if

$$\llbracket \neg\varphi \mathbin{?} \Psi_2 : \Psi_1 \rrbracket (w,\sigma) = \llbracket \Psi_1 \rrbracket (w,\sigma),$$

and likewise $\llbracket \varphi \mathbin{?} \Psi_1 : \Psi_2 \rrbracket (w,\sigma) = \llbracket \Psi_2 \rrbracket (w,\Sigma)$ if and only if

$$\llbracket \neg\varphi \mathbin{?} \Psi_2 : \Psi_1 \rrbracket (w,\sigma) = \llbracket \Psi_2 \rrbracket (w,\sigma).$$

It follows that $\varphi \mathbin{?} \Psi_1 : \Psi_2 \sim_\Gamma \neg\varphi \mathbin{?} \Psi_2 : \Psi_1$.

$(S3)$: Assume $\Gamma \vdash \varphi$. By Corollary 3, this means that $\Gamma \models \varphi$. Hence, for any $(w,\sigma) \in \llbracket \Gamma \rrbracket$ we have $(w,\sigma) \models \varphi$, so $\varphi \mathbin{?} \Psi_1 : \Psi_2 \sim_\Gamma \Psi_1$.

$(S4)$: Assume that $\Psi_1 \sim_{\Gamma\cup\{\varphi_1,\varphi_2\}} \Psi_1'$, $\Psi_1 \sim_{\Gamma\cup\{\varphi_1,\neg\varphi_2\}} \Psi_2'$, $\Psi_2 \sim_{\Gamma\cup\{\neg\varphi_1,\varphi_2\}} \Psi_1'$, and $\Psi_2 \sim_{\Gamma\cup\{\neg\varphi_1,\neg\varphi_2\}} \Psi_2'$. Let $(w,\sigma) \in \llbracket \Gamma \rrbracket$. Since $\llbracket \{\varphi_1,\varphi_2\} \rrbracket$, $\llbracket \{\varphi_1,\neg\varphi_2\} \rrbracket$, $\llbracket \{\neg\varphi_1,\varphi_2\} \rrbracket$, and $\llbracket \{\neg\varphi_1,\neg\varphi_2\} \rrbracket$ partition Σ_σ^+, (w,σ) must be in one of these sets. Accordingly, we can conclude that

$$\llbracket \varphi_1 \mathbin{?} \Psi_1 : \Psi_2 \rrbracket (w,\sigma) = \llbracket \varphi_2 \mathbin{?} \Psi_1' : \Psi_2' \rrbracket (w,\sigma)$$

from the corresponding assumption, so $\varphi_1 \mathbin{?} \Psi_1 : \Psi_2 \sim_\Gamma \varphi_2 \mathbin{?} \Psi_1' : \Psi_2'$. ◀

▶ **Proposition 27.** *The following theorems can be derived in core-wMSO$(?,+)$.*

1. $\Gamma \vdash \Phi_1 \approx \Phi_2$ *for any* Φ_1 *and* Φ_2 *if* Γ *is inconsistent.*
2. $\Gamma \vdash \Phi_1 \approx \Phi_2$ *if* $\Gamma \vdash \varphi$ *and* $\Gamma \cup \{\varphi\} \vdash \Phi_1 \approx \Phi_2$.
3. $\Gamma \vdash \varphi \mathbin{?} \Phi : \Phi \approx \Phi$.
4. *If* $\Gamma \cup \{\varphi_1,\varphi_2\} \vdash \Phi_1 \approx \Phi_1'$, $\Gamma \cup \{\varphi_1,\neg\varphi_2\} \vdash \Phi_1 \approx \Phi_2'$, $\Gamma \cup \{\neg\varphi_1,\varphi_2\} \vdash \Phi_2 \approx \Phi_1'$, *and* $\Gamma \cup \{\neg\varphi_1,\neg\varphi_2\} \vdash \Phi_2 \approx \Phi_2'$, *then* $\Gamma \vdash \varphi_1 \mathbin{?} \Phi_1 : \Phi_2 \approx \varphi_2 \mathbin{?} \Phi_1' : \Phi_2'$.
5. $\Gamma \vdash \varphi_1 \mathbin{?} \Phi_1 : \Phi_1 \approx \varphi_2 \mathbin{?} \Phi_1 : \Phi_2$ *if* $\Gamma \vdash \varphi_1 \leftrightarrow \varphi_2$.
6. $\Gamma \vdash \varphi \mathbin{?} \Phi_1 : \Phi_2 \approx \Phi_2$ *if* $\Gamma \vdash \neg\varphi$.
7. *If* $\Gamma \cup \{\varphi\} \vdash \Phi_1 \approx \Phi_1'$ *and* $\Gamma \cup \{\neg\varphi\} \vdash \Phi_2 \approx \Phi_2'$ *then* $\Gamma \vdash \varphi \mathbin{?} \Phi_1 : \Phi_2 \approx \varphi \mathbin{?} \Phi_1' : \Phi_2'$.
8. *If* $\Gamma \cup \{\varphi\} \vdash \Phi_1 \approx \Phi_2$ *and* $\Gamma \cup \{\neg\varphi\} \vdash \Phi_1 \approx \Phi_2$ *then* $\Gamma \vdash \Phi_1 \approx \Phi_2$.
9. $\Gamma \cup \{\varphi\} \vdash \varphi \mathbin{?} \Phi_1 : \Phi_2 \approx \Phi_1$.

Proof. Similar to the proof of Proposition 4. ◀

Proof of Lemma 7. $(\implies)$ $\prod_x \Psi_1 \sim_\Gamma \prod_x \Psi_2$ implies that $\llbracket \Psi_1 \rrbracket (w,\sigma[x \mapsto i]) = \llbracket \Psi_2 \rrbracket (w,\sigma[x \mapsto i])$ for all i and (w,σ) such that $(w,\sigma) \models \Gamma$. This implies $\llbracket \Psi_1 \rrbracket (w,\sigma) = \llbracket \Psi_2 \rrbracket (w,\sigma)$ for all $(w,\sigma) \models \forall x(\Gamma)$, so $\Psi_1 \sim_{\forall x(\Gamma)} \Psi_2$.

$(\impliedby)$ $\Psi_1 \sim_{\forall x(\Gamma)} \Psi_2$ means that $\llbracket \Psi_1 \rrbracket (w,\sigma) = \llbracket \Psi_2 \rrbracket (w,\sigma)$ for all $(w,\sigma) \models \forall x(\Gamma)$. This implies that $\llbracket \Psi_1 \rrbracket (w,\sigma[x \mapsto i]) = \llbracket \Psi_2 \rrbracket (w,\sigma[x \mapsto i])$ for all i and $(w,\sigma) \models \Gamma$. This in turn implies $\llbracket \prod_x \Psi_1 \rrbracket (w,\sigma) = \llbracket \prod_x \Psi_2 \rrbracket (w,\sigma)$ for all $(w,\sigma) \models \Gamma$, so $\prod_x \Psi_1 \sim_\Gamma \prod_x \Psi_2$. ◀

▶ **Lemma 28.** *If* Φ_1 *and* Φ_2 *are in normal form, then there exists a formula* Φ*, also in normal form, such that* $\Gamma \vdash \Phi \approx \Phi_1 + \Phi_2$.

Proof. The proof is by induction on the maximum number of nested occurences of the conditional operator within Φ_1 and Φ_2. Note that since these are in normal form, occurences of the conditional operator will always occur consecutively as the outermost operators. Formally, we define, on formulas Φ in normal form, the following function which counts the number of nested occurences of the conditional operator:

$$\#?(\Phi) = \begin{cases} 1 + \max\{\#?(\Phi'), \#?(\Phi'')\} & \text{if } \Phi = \varphi \mathbin{?} \Phi' : \Phi'' \\ 0 & \text{otherwise.} \end{cases}$$

Let $k = \max\{\#?(\Phi_1), \#?(\Phi_2)\}$.

$k = 0$: If $\Phi_1 = \mathbf{0}$, then $\Gamma \vdash \Phi_1 + \Phi_2 \approx \Phi_2$ by $(C1)$, and since Φ_2 was assumed to be in normal form, we can take $\Phi = \Phi_2$. Similarly if $\Phi_2 = \mathbf{0}$. If neither $\Phi_1 = \mathbf{0}$ or $\Phi_2 = \mathbf{0}$, then $\Phi_1 + \Phi_2$ is already in normal form, so we can simply take $\Phi = \Phi_1 + \Phi_2$.

$k > 0$: We have three cases to consider:

(1) $\#?(\Phi_1) = \#?(\Phi_2)$,

(2) $\#?(\Phi_1) < \#?(\Phi_2)$, or

(3) $\#?(\Phi_1) > \#?(\Phi_2)$.

(1) Consider $\Phi_1 = \varphi_1 ? \Phi_1' : \Phi_1''$ and $\Phi_2 = \varphi_2 ? \Phi_2' : \Phi_2''$. Now, by three applications of axiom $(C10)$, we get

$$\Gamma \vdash \varphi_1 ? \Phi_1' : \Phi_1'' + \varphi_2 ? \Phi_2' : \Phi_2'' \approx \varphi_1 ? (\varphi_2 ? \Phi_1' + \Phi_2' : \Phi_1' + \Phi_2'') : (\varphi_2 ? \Phi_1'' + \Phi_2' : \Phi_1'' + \Phi_2'').$$

Since $\#?(\Phi_1' + \Phi_2') < k$, $\#?(\Phi_1' + \Phi_2'') < k$, $\#?(\Phi_1'' + \Phi_2') < k$, and $\#?(\Phi_1'' + \Phi_2'') < k$, the induction hypothesis gives formulas Φ', Φ'', Φ''', and Φ'''', all in normal form, such that $\Gamma \vdash \Phi' \approx \Phi_1' + \Phi_2'$, $\Gamma \vdash \Phi'' \approx \Phi_1' + \Phi_2''$, $\Gamma \vdash \Phi''' \approx \Phi_1'' + \Phi_2'$, and $\Gamma \vdash \Phi'''' \approx \Phi_1'' + \Phi_2''$. Thus

$$\Phi = \varphi_1 ? (\varphi_2 ? \Phi' : \Phi'') : (\varphi_2 ? \Phi''' : \Phi'''')$$

is in normal form and satisfies $\Gamma \vdash \Phi \approx \Phi_1 + \Phi_2$.

(2), (3) These cases are simpler versions of case (1). ◀

Proof of Lemma 9. The proof is by induction on the structure of Φ.

$\Phi = 0$ or $\Phi = \prod_x \Psi$: In this case Φ is already in normal form.

$\Phi = \Phi_1 + \Phi_2$: By induction hypothesis, there exist formulas Φ_1' and Φ_2', both in normal form, such that $\Gamma \vdash \Phi_1 \approx \Phi_1'$ and $\Gamma \vdash \Phi_2 \approx \Phi_2'$. By Lemma 28, there exists a formula Φ' in normal form such that $\Gamma \vdash \Phi' \approx \Phi_1' + \Phi_2'$. By congruence we get $\Gamma \vdash \Phi_1 + \Phi_2 \approx \Phi_1' + \Phi_2'$, so $\Gamma \vdash \Phi \approx \Phi'$.

$\Phi = \varphi ? \Phi_1 : \Phi_2$: By induction hypothesis there exist Φ_1' and Φ_2' in normal form such that $\Gamma \vdash \Phi_1 \approx \Phi_1'$ and $\Gamma \vdash \Phi_2 \approx \Phi_2'$. Then $\Phi' = \varphi ? \Phi_1' : \Phi_2'$ is in normal form and, by congruence, $\Gamma \vdash \Phi \approx \Phi'$. ◀

Proof of Lemma 10. We show why the first equivalence is true; the remaining cases are similar. Let $\Phi_1 = \varphi_1 ? \Phi_1' : \Phi_1''$ and $\Phi_2 = \varphi_2 ? \Phi_2' : \Phi_2''$. If $(w, \sigma) \in [\![\Gamma \cup \{\varphi_1, \varphi_2\}]\!]$, then also $(w, \sigma) \in [\![\Gamma]\!]$, so

$$[\![\Phi_1']\!](w, \sigma) = [\![\Phi_1]\!](w, \sigma) = [\![\Phi_2]\!](w, \sigma) = [\![\Phi_2']\!](w, \sigma). \quad ◀$$

▶ **Lemma 29.** *Given two formulas Ψ_1 and Ψ_2, there exists a formula φ_{Ψ_1,Ψ_2} such that $(w, \sigma) \models \forall x.\varphi_{\Psi_1,\Psi_2}$ if and only if $[\![\prod_x \Psi_1]\!](w, \sigma) = [\![\prod_x \Psi_2]\!](w, \sigma)$. In particular,*

$$\prod_x \Psi_1 \sim_{\Gamma \cup \{\forall x.\varphi_{\Psi_1,\Psi_2}\}} \prod_x \Psi_2.$$

Proof. Consider the sets R_1 and R_2 of values that appear in Ψ_1 and Ψ_2, respectively. If these sets are disjoint, then $[\![\prod_x \Psi_1]\!](w, \sigma) \neq [\![\prod_x \Psi_2]\!](w, \sigma)$ for all (w, σ), so we can take $\varphi_{\Psi_1,\Psi_2} = \bot$.

If they are not disjoint, consider some value $r \in R_1 \cap R_2$. Then there must exist a formula $\varphi^r_{\Psi_1}$ such that $(w, \sigma) \models \varphi^r_{\Psi_1}$ if and only if $[\![\Psi_1]\!](w, \sigma) = r$. Likewise, there exists $\varphi^r_{\Psi_2}$ such that $(w, \sigma) \models \varphi^r_{\Psi_2}$ if and only if $[\![\Psi_2]\!](w, \sigma) = r$. Now we take $\varphi^r_{\Psi_1,\Psi_2} = \varphi^r_{\Psi_1} \wedge \varphi^r_{\Psi_2}$ and

$$\varphi_{\Psi_1,\Psi_2} = \bigvee_{r \in R_1 \cap R_2} \varphi^r_{\Psi_1,\Psi_2}.$$

We now have a formula φ_{Ψ_1,Ψ_2} such that, for all (w,σ), $(w,\sigma) \models \varphi_{\Psi_1,\Psi_2}$ if and only if $[\![\Psi_1]\!](w,\sigma) = [\![\Psi_2]\!](w,\sigma)$. This is equivalent to

$$\forall (w,\sigma).\forall i \in \{1,\ldots,|w|\}.(w,\sigma[x\mapsto i]) \models \varphi_{\Psi_1,\Psi_2} \text{ iff } [\![\Psi_1]\!](w,\sigma[x\mapsto i]) = [\![\Psi_2]\!](w,\sigma[x\mapsto i])$$

which implies that

$$\forall (w,\sigma).(w,\sigma) \models \forall x.\varphi_{\Psi_1,\Psi_2} \text{ iff } [\![\textstyle\prod_x \Psi_1]\!](w,\sigma) = [\![\textstyle\prod_x \Psi_2]\!](w,\sigma). \qquad \blacktriangleleft$$

▶ **Lemma 30.** *If $\Gamma \vdash \bigvee_{m=1}^{n} \varphi_m$ and for all m it holds that $\Gamma \cup \{\varphi_m\} \vdash \Phi_1 \approx \Phi_2$, then $\Gamma \vdash \Phi_1 \approx \Phi_2$.*

Proof. The proof is by induction on n. The case of $n = 1$ is trivial: we have assumed that $\Gamma \cup \{\varphi_1\} \vdash \Phi_1 \approx \Phi_2$, and therefore Proposition 27(2) gives $\Gamma \vdash \Phi_1 \approx \Phi_2$. Now, let $n = k+1$. We have that $\Gamma \cup \{\varphi_n\} \vdash \Phi_1 \approx \Phi_2$ and that $\Gamma \cup \{\neg\varphi_n\} \vdash \bigvee_{m=1}^{k} \varphi_m$, so by the inductive hypothesis, $\Gamma \cup \{\neg\varphi_n\} \vdash \Phi_1 \approx \Phi_2$. Therefore, from Proposition 27(8), we have that $\Gamma \vdash \Phi_1 \approx \Phi_2$. ◀

Proof of Lemma 11. By definition, $\Phi_1 \sim_\Gamma \Phi_2$ means that for all $(w,\sigma) \in [\![\Gamma]\!]$ there exists a permutation $(j_1,\ldots,j_k)$ such that

$$[\![\textstyle\prod_x \Psi_i]\!](w,\sigma) = [\![\textstyle\prod_x \Psi'_{j_i}]\!](w,\sigma) \tag{1}$$

for all i.

By Lemma 29, for each such permutation $P = (j_1,\ldots,j_k)$ there exist formulas $\varphi_{1,j_1},\ldots,\varphi_{k,j_k}$ such that

$$\textstyle\prod_x \Psi_i \sim_{\Gamma\cup\{\forall x.\varphi_{i,j_i}\}} \prod_x \Psi'_{j_i},$$

and by assumption, this gives

$$\Gamma \cup \{\forall x.\varphi_{i,j_i}\} \vdash \textstyle\prod_x \Psi_i \approx \prod_x \Psi'_{j_i}. \tag{2}$$

For each permutation $P = \{j_1,\ldots,j_k\}$, let

$$\varphi_P = (\forall x.\varphi_{1,j_1}) \wedge \cdots \wedge (\forall x.\varphi_{k,j_k}).$$

By Equation (1) and Lemma 29, for every $(w,\sigma) \in [\![\Gamma]\!]$ there exists a permutation $P = (j_1,\ldots,j_k)$ such that we have $(w,\sigma) \models \varphi_P$. This means that for all $(w,\sigma) \in [\![\Gamma]\!]$ we have $(w,\sigma) \models \bigvee_P \varphi_P$. By Corollary 3, this means that $\Gamma \vdash \bigvee_P \varphi_P$.

Now, from Equation (2), we can use $(C6)$ to get

$$\Gamma \cup \{\varphi_P\} \cup \{\forall x.\varphi_{i,j_i}\} \vdash \textstyle\prod_x \Psi_i \approx \prod_x \Psi'_{j_i},$$

and together with $\Gamma \cup \{\varphi_P\} \vdash \forall x.\varphi_{i,j_i}$, this gives $\Gamma \cup \{\varphi_P\} \vdash \prod_x \Psi_i \approx \prod_x \Psi'_{j_i}$ by Proposition 27(2). We can then use congruence to get

$$\Gamma \cup \{\varphi_P\} \vdash \sum_{i=1}^{k} \textstyle\prod_x \Psi_i \approx \displaystyle\sum_{j=1}^{k} \textstyle\prod_x \Psi'_{j_i}. \tag{3}$$

Since $\sum_{j=1}^{k} \prod_x \Psi'_{j_i}$ is a permutation of Φ_2, we get by axioms $(C2)$ and $(C3)$ that $\Gamma \cup \{\varphi_P\} \vdash \Phi_2 \approx \sum_{j=1}^{k} \prod_x \Psi'_{j_i}$, so

$$\Gamma \cup \{\varphi_P\} \vdash \Phi_1 \approx \Phi_2 \tag{4}$$

by Equation (3). By Lemma 30, Equation (4) together with the fact that $\Gamma \vdash \bigvee_P \varphi_P$ gives

$$\Gamma \vdash \Phi_1 \approx \Phi_2. \qquad \blacktriangleleft$$

Proof of Lemma 12. If Γ is inconsistent, then $\Gamma \vdash \Phi_1 \approx \Phi_2$ always holds, so we are done. Assume therefore that Γ is consistent. The proof now proceeds by induction on the maximum of $\mathtt{depth}(\Phi_1)$ and $\mathtt{depth}(\Phi_2)$, where

$$\mathtt{depth}(\Phi) = \begin{cases} 0 & \text{if } \Phi = \mathbf{0} \text{ or } \Phi = \prod_x \Psi \\ 1 + \max\{\mathtt{depth}(\Phi'), \mathtt{depth}(\Phi'')\} & \text{if } \Phi = \varphi \mathbin{?} \Phi' : \Phi'' \text{ or } \Phi = \Phi' + \Phi'' \end{cases}$$

Case $\max\{\mathtt{depth}(\Phi_1), \mathtt{depth}(\Phi_2)\} = 0$: Since $\Phi_1 \sim_\Gamma \Phi_2$, we must have $\Phi_1 = \mathbf{0} = \Phi_2$ or $\Phi_1 = \prod_{x_1} \Psi_1$ and $\Phi_2 = \prod_{x_2} \Psi_2$. In the first case, $\Gamma \vdash \mathbf{0} \approx \mathbf{0}$ by reflexivity. In the second case, we can find some $x \notin \mathtt{var}(\Psi_1) \cup \mathtt{var}(\Psi_2)$ such that $\prod_x \Psi_1[x/x_1] \sim_\Gamma \prod_x \Psi_2[x/x_2]$. By Lemma 7 we get $\Psi_1[x/x_1] \sim_{\forall x(\Gamma)} \Psi_2[x/x_2]$, and by completeness of step-wMSO, this implies $\forall x(\Gamma) \vdash \Psi_1[x/x_1] \approx \Psi_2[x/x_2]$. We can then use axiom $(C4)$ to obtain $\Gamma \vdash \prod_x \Psi_1[x/x_1] \approx \prod_x \Psi_2[x/x_2]$, and finally use axiom $(C5)$ to obtain $\Gamma \vdash \prod_{x_1} \Psi_1 \approx \prod_{x_2} \Psi_2$.

Case $\max\{\mathtt{depth}(\Phi_1), \mathtt{depth}(\Phi_2)\} > 0$:

Case $\Phi_1 = \Phi_1' + \Phi_1''$: In this case we have one of the following sub-cases:

(1) $\Phi_2 = \mathbf{0}$,
(2) $\Phi_2 = \prod_x \Psi$,
(3) $\Phi_2 = \varphi_2 \mathbin{?} \Phi_2' : \Phi_2''$, or
(4) $\Phi_2 = \Phi_2' + \Phi_2''$.

(1) This case can not happen, since we must have $\Phi_1' = \mathbf{0} = \Phi_1''$, but this is not allowed in normal form.

(2) This can not happen, since $|\llbracket \Phi_1 \rrbracket (w, \sigma)| \geq 2$ but $|\llbracket \Phi_2 \rrbracket (w, \sigma)| = 1$, which contradicts $\Phi_1 \sim_\Gamma \Phi_2$.

(3) Note first that we may have $\Phi_2' = \mathbf{0}$ or $\Phi_2'' = \mathbf{0}$. If $\Phi_2' = \mathbf{0}$, then we must have $\Gamma \vdash \neg\varphi_2$, because otherwise there would exist $(w, \sigma) \in \llbracket \Gamma \rrbracket$ such that $\llbracket \Phi_2 \rrbracket (w, \sigma) = \llbracket \Phi_2'' \rrbracket (w, \sigma) = \emptyset$, which contradicts $\Phi_1 \sim_\Gamma \Phi_2$. Then $\Gamma \vdash \varphi_2 \mathbin{?} \Phi_2' : \Phi_2'' \approx \Phi_2''$ by Proposition 27(4), and since $\Phi_1 \sim_\Gamma \Phi_2''$, Lemma 11 gives $\Gamma \vdash \Phi_1 \approx \Phi_2''$, so $\Gamma \vdash \Phi_2 \approx \Phi_2$. Likewise if $\Phi_2'' = \mathbf{0}$.
Assume now that $\Phi_2' \neq \mathbf{0}$ and $\Phi_2'' \neq \mathbf{0}$. We proceed by induction on $\#?(\Phi_2)$.
If $\#?(\Phi_2) = 1$, then we have $\Phi_1 \sim_{\Gamma \cup \{\varphi_2\}} \Phi_2'$ and $\Phi_1 \sim_{\Gamma \cup \{\neg\varphi_2\}} \Phi_2''$, and since Φ_2 is in normal form, Lemma 11 gives $\Gamma \cup \{\varphi_2\} \vdash \Phi_1 \approx \Phi_2'$ and $\Gamma \cup \{\neg\varphi_2\} \vdash \Phi_1 \approx \Phi_2''$, so $\Gamma \vdash \Phi_1 \approx \Phi_2$ by $(C9)$.
If $\#?(\Phi_2) > 1$, then $\Phi_1 \sim_{\Gamma \cup \{\varphi_2\}} \Phi_2'$ and $\Phi_1 \sim_{\Gamma \cup \{\neg\varphi_2\}} \Phi_2''$, so the induction hypothesis gives $\Gamma \cup \{\varphi_2\} \vdash \Phi_1 \approx \Phi_2'$ and $\Gamma \cup \{\neg\varphi_2\} \vdash \Phi_1 \approx \Phi_2''$, so we conclude $\Gamma \vdash \Phi_1 \approx \Phi_2$ by $(C9)$.

(4) Since Φ_1 and Φ_2 are in normal form, we must have $\Phi_1 = \sum_{i=1}^{k} \prod_x \Psi_i$ and $\Phi_2 = \sum_{j=1}^{k'} \prod_x \Psi_j'$, and furthermore we must have $k = k'$ since otherwise $|\llbracket \Phi_1 \rrbracket (w, \sigma)| \neq |\llbracket \Phi_2 \rrbracket (w, \sigma)|$, contradicting $\Phi_1 \sim_\Gamma \Phi_2$. By the induction hypothesis, $\prod_x \Psi_i \sim_\Gamma \prod_x \Psi_j'$ implies $\Gamma \vdash \prod_x \Psi_i \approx \prod_x \Psi_j'$, so Lemma 11 gives $\Gamma \vdash \Phi_1 \approx \Phi_2$.

Case $\Phi_1 = \varphi_1 \mathbin{?} \Phi_1' : \Phi_1''$: In this case we have either

(1) $\Phi_2 = \mathbf{0}$,
(2) $\Phi_2 = \prod_x \Psi$,
(3) $\Phi_2 = \varphi_2 \mathbin{?} \Phi_2' : \Phi_2''$, or
(4) $\Phi_2 = \Phi_2' + \Phi_2''$.

(1), (2) In both cases we have $\Phi_1' \sim_{\Gamma \cup \{\varphi_1\}} \Phi_2$ and $\Phi_1'' \sim_{\Gamma \cup \{\neg\varphi_1\}} \Phi_2$, so $\Gamma \cup \{\varphi_1\} \vdash \Phi_1' \approx \Phi_2$ and $\Gamma \cup \{\neg\varphi_1\} \vdash \Phi_1'' \approx \Phi_2$ by induction hypothesis. Then $(C9)$ gives $\Gamma \vdash \Phi_1 \approx \Phi_2$.

(3) By Lemma 10, we know that

$$\begin{array}{ll}\Phi_1' \sim_{\Gamma\cup\{\varphi_1,\varphi_2\}} \Phi_2', & \Phi_1' \sim_{\Gamma\cup\{\varphi_1,\neg\varphi_2\}} \Phi_2'', \\ \Phi_1'' \sim_{\Gamma\cup\{\neg\varphi_1,\varphi_2\}} \Phi_2', \text{ and} & \Phi_1'' \sim_{\Gamma\cup\{\neg\varphi_1,\neg\varphi_2\}} \Phi_2''.\end{array}$$

The induction hypothesis then gives

$$\begin{array}{ll}\Gamma\cup\{\varphi_1,\varphi_2\} \vdash \Phi_1' \approx \Phi_2', & \Gamma\cup\{\varphi_1,\neg\varphi_2\} \vdash \Phi_1' \approx \Phi_2'', \\ \Gamma\cup\{\neg\varphi_1,\varphi_2\} \vdash \Phi_1'' \approx \Phi_2', \text{ and} & \Gamma\cup\{\neg\varphi_1,\neg\varphi_2\} \vdash \Phi_1'' \approx \Phi_2'',\end{array}$$

so all the assumptions for Proposition 27(2) are met, and we conclude $\Gamma \vdash \Phi_1 \approx \Phi_2$.

(4) This case is symmetric to case $\Phi_1 = \Phi_1' + \Phi_1''$(3). ◀

Proof of Theorem 13. ($\Longleftarrow$): We show the soundness of each axiom in turn.

($C1$):

$$[\![\Phi + \mathbf{0}]\!](w,\sigma) = [\![\Phi]\!](w,\sigma) \uplus [\![\mathbf{0}]\!](w,\sigma) = [\![\Phi]\!](w,\sigma) \uplus \emptyset = [\![\Phi]\!](w,\sigma).$$

($C2$):

$$[\![\Phi_1 + \Phi_2]\!](w,\sigma) = [\![\Phi_1]\!](w,\sigma) \uplus [\![\Phi_2]\!](w,\sigma) = [\![\Phi_2]\!](w,\sigma) \uplus [\![\Phi_1]\!](w,\sigma) = [\![\Phi_2 + \Phi_1]\!](w,\sigma).$$

($C3$):

$$\begin{aligned}[\![(\Phi_1 + \Phi_2) + \Phi_3]\!](w,\sigma) &= [\![(\Phi_1 + \Phi_2)]\!](w,\sigma) \uplus [\![\Phi_3]\!](w,\sigma) \\ &= ([\![\Phi_1]\!](w,\sigma) \uplus [\![\Phi_2(w,\sigma)]\!]) \uplus [\![\Phi_3]\!](w,\sigma) \\ &= [\![\Phi_1]\!](w,\sigma) \uplus ([\![\Phi_2]\!](w,\sigma) \uplus [\![\Phi_3]\!](w,\sigma)) \\ &= [\![\Phi_1]\!](w,\sigma) \uplus [\![\Phi_2 + \Phi_3]\!](w,\sigma) \\ &= [\![\Phi_1 + (\Phi_2 + \Phi_3)]\!](w,\sigma).\end{aligned}$$

($C4$): This follows from soundness of step-wMSO and Lemma 7.

($C5$): If $y \notin \mathtt{var}(\Psi)$, then

$$\begin{aligned}[\![\textstyle\prod_x \Psi]\!](w,\sigma) &= \{\!|\, [\![\Psi]\!](w,\sigma[x \mapsto 1]\ldots[\![\Psi]\!](w,\sigma[x \mapsto |w|)\,|\!\} \\ &= \{\!|\, [\![\Psi[y/x]]\!](w,\sigma[y \mapsto 1])\ldots[\![\Psi[y/x]]\!](w,\sigma[y \mapsto |w|])\,|\!\} \\ &= \left[\!\!\left[\textstyle\prod_y \Psi[y/x]\right]\!\!\right](w,\sigma).\end{aligned}$$

($C6$)–($C9$): The proof of these is similar to the corresponding proofs in Theorem 6.

($C10$): We evaluate by cases. If $(w,\sigma) \models \varphi$, then

$$\begin{aligned}[\![(\varphi\,?\,\Phi' : \Phi'') + \Phi]\!](w,\sigma) &= [\![(\varphi\,?\,\Phi' : \Phi'')]\!](w,\sigma) \uplus [\![\Phi]\!](w,\sigma) \\ &= [\![\Phi']\!](w,\sigma) \uplus [\![\Phi]\!](w,\sigma) \\ &= [\![\Phi' + \Phi]\!](w,\sigma).\end{aligned}$$

and

$$[\![\varphi\,?\,\Phi' + \Phi : \Phi'' + \Phi]\!](w,\sigma) = [\![\Phi' + \Phi]\!](w,\sigma)$$

Likewise, if $(w,\sigma) \models \neg\varphi$, then

$$\begin{aligned}[\![(\varphi\,?\,\Phi' : \Phi'') + \Phi]\!](w,\sigma) &= [\![(\varphi\,?\,\Phi' : \Phi'')]\!](w,\sigma) \uplus [\![\Phi]\!](w,\sigma) \\ &= [\![\Phi'']\!](w,\sigma) \uplus [\![\Phi]\!](w,\sigma) \\ &= [\![\Phi'' + \Phi]\!](w,\sigma)\end{aligned}$$

and

$$[\![\varphi\,?\,\Phi' + \Phi : \Phi'' + \Phi]\!](w,\sigma) = [\![\Phi'' + \Phi]\!](w,\sigma).$$

($\implies$): Assume $\Phi_1 \sim_\Gamma \Phi_2$. By Lemma 9, there exist formulas Φ_1' and Φ_2', both in normal form, such that $\Gamma \vdash \Phi_1 \approx \Phi_1'$ and $\Gamma \vdash \Phi_2 \approx \Phi_2'$. By soundness, this implies $\Phi_1 \sim_\Gamma \Phi_1'$ and $\Phi_2 \sim_\Gamma \Phi_2'$, so $\Phi_1' \sim_\Gamma \Phi_2'$. Since these are in normal form, Lemma 12 gives $\Gamma \vdash \Phi_1' \approx \Phi_2'$, and by symmetry and transitivity, this implies $\Gamma \vdash \Phi_1 \approx \Phi_2$. ◀

Proof of Lemma 24. The lemma results from observing that the elements of the multisets $[\![\Phi_1 + \Phi_1']\!]\,(w,\sigma)$ and $[\![\Phi_2 + \Phi_2']\!]\,(w,\sigma)$ can be partitioned into those that use values that appear in Φ_1 and Φ_2, and those that use values that appear in Φ_1' and Φ_2'. ◀

B The Full Proof for the Undecidability of Equational Satisfiability

We now present the full construction of the reduction that proves that equational satisfiability of core-wFO (and therefore also of core-wMSO) is undecidable. We take special care to only use core-wFO formulas, and therefore we use a special construction for recording the positions where each symbol appears in a configuration.

Fix a pair (w,σ). We use a series of formulas and equations to express that a (w,σ) encodes the computation of a Turing Machine that halts. Therefore, the question of whether there is such a pair that satisfies the resulting set of equations is undecidable. Let $T = (Q, \Sigma, \delta, q_0, H)$ be a Turing Machine, where Q is a finite set of states, Σ is the set of symbols that the machine uses, $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, R\}$ is the machine's transition function, q_0 is the starting state, and H is the halting state of T. Let $\lhd, \mathtt{m}, \mathtt{1}$ be special symbols not in Σ. A configuration of T is represented by a string of the form $s_1 q s_2 \lhd$, where q is the current state for the configuration, $s_1 s_2$ is the string of symbols in the tape of the machine, and the head is located at the first symbol of s_2; $\lhd$ marks the end of the configuration. Let $x_0 \in \Sigma^*$ be an input of T.

We use every $s \in Q \cup \Sigma \cup \{\lhd, \mathtt{1}, \mathtt{m}\}$ as a predicate, so that $s(x)$ is true if and only if the symbol s is in position x. Let $[0] = \mathtt{1}$, and for every $i \geq 1$, let $[i] = \mathtt{1}^{2^{i-1}}\mathtt{m}\mathtt{1}^{2^{i-1}}$, so that in $[i]$, $\mathtt{1}$ appears exactly 2^i times. Then, for every string $y_0 y_1 \cdots y_j \in (Q \cup \Sigma \cup \{\lhd\})^j$, let $[y_0 y_1 \cdots y_j] = [0] y_0 [1] y_1 \cdots [j] y_j$. We want to describe that (w,σ) encodes a halting run of T on x_0. In other words, we must ensure that (w,σ) is $[c_0] \cdots [c_k]$, where $c_0 \cdots c_k$ is a sequence of configurations of T, such that c_0 is $q_0 x_0 \lhd$ and c_k is $s_1 H s_2 \lhd$, where $s_1, s_2 \in \Sigma^*$.

We must therefore ensure that the following conditions hold:

1. (w,σ) is of the form $[c_0][c_1] \cdots [c_k]$, where each c_i has exactly one $\lhd$, at the end;
2. each c_i is of the form $s_1 q s_2 \lhd$, where $q \in Q$, $s_1 s_2 \in \Sigma^*$, and $s_2 \neq \varepsilon$;
3. $c_0 = q_0 x_0 \lhd$;
4. $c_k = s_1 H s_2 \lhd$ for some s_1, s_2; and
5. for every $0 \leq i < k$, c_{i+1} results from c_i by applying the transition function δ. This condition can be further refined into the following subconditions. For every $0 \leq i < k$, if $c_i = x_1\ x_2 \cdots\ x_r\ q_i\ y_1\ y_2 \cdots\ y_{r'} \lhd$, then:
   a. if $\delta(q_i, y_1) = (q, x, L)$ and $r > 0$, then $c_{i+1} = x_1\ x_2 \cdots\ x_{r-1}\ q\ x_r\ x\ y_2 \cdots\ y_{r'} \lhd$,
   b. if $\delta(q_i, y_1) = (q, x, L)$ and $r = 0$, then $c_{i+1} = q\ x\ y_2 \cdots\ y_{r'} \lhd$,
   c. if $\delta(q_i, y_1) = (q, x, R)$ and $r' > 1$, then $c_{i+1} = x_1\ x_2 \cdots\ x_r\ x\ q\ y_2 \cdots\ y_{r'} \lhd$, and
   d. if $\delta(q_i, y_1) = (q, x, R)$ and $r' = 1$, then $c_{i+1} = x_1\ x_2 \cdots\ x_r\ x\ q\ \_\lhd$, where $\_ \in \Sigma$ is the symbol used by T for a blank space.

We now explain how to represent each of the conditions above with a formula or equation. We use the following macros, where $0, 1 \in R$ are two distinct weights:

$$\begin{aligned}
\texttt{smbl}(x) &\stackrel{\text{def}}{=} \bigvee_{s \in Q \cup \Sigma \cup \{\triangleleft\}} s(x) \\
\texttt{nxt}(x,y) &\stackrel{\text{def}}{=} (\neg(y \leq x)) \wedge \forall z.\ (z \leq x \vee y \leq z) \\
\texttt{first}(x) &\stackrel{\text{def}}{=} \forall y.\ y \geq x \quad \text{and} \quad \texttt{first-cf}(x) \stackrel{\text{def}}{=} \texttt{first}(x) \vee \exists y.\ \triangleleft(y) \wedge \texttt{nxt}(y,x) \\
\texttt{last}(x) &\stackrel{\text{def}}{=} \forall y.\ y \leq x \\
\texttt{nxt-sm}(x,y) &\stackrel{\text{def}}{=} (\neg y \leq x) \wedge \forall z.\ (\neg\texttt{smbl}(z) \vee z \leq x \vee y \leq z) \\
v_1(x) &\stackrel{\text{def}}{=} \prod_y (x = y)\ ?\ 1 : 0, \quad v_s^x \stackrel{\text{def}}{=} \prod_y (x = y)\ ?\ s : 0, \quad \text{and} \quad v_0 \stackrel{\text{def}}{=} \mathbf{0} \\
\texttt{ps}_v(x) &\stackrel{\text{def}}{=} \forall y.\ \neg\texttt{smbl}(y) \vee x \leq y\ ?\ v : \\
&\qquad \sum_y \exists z.\ \texttt{smbl}(z) \wedge \texttt{nxt-sm}(z,x) \wedge z \leq y \leq x \wedge \texttt{1}(y)\ ?\ v : v_0
\end{aligned}$$

Intuitively, formula $\texttt{ps}_v(x)$ counts how many 1s appear right before position x. We note that, as long as condition 1 is satisfied, for each symbol s that appears in the set S of positions in a configuration, S is uniquely identified by $\sum_{i \in S} 2^i$. Furthermore, for each configuration, $\texttt{ps}_v(x)$ constructs a map from each such s (represented by the returned value v) to $\sum_{i \in S} 2^i$. Therefore, the way that we will use $\texttt{ps}_v(x)$ (see how we deal with condition 5, below) gives a complete description of each configuration.

We will use v_0 as the default (negative) value in conditionals, and as such $\varphi\ ?\ v$ is used as shorthand for $\varphi\ ?\ v : v_0$. Furthermore, we assume that : binds to the nearest ?, and therefore, $\varphi_1\ ?\ \varphi_2\ ?\ \Phi_1 : \Phi_2$ means $\varphi_1\ ?\ \varphi_2\ ?\ \Phi_1 : \Phi_2 : v_0$, which can be uniquely parsed as $\varphi_1\ ?\ (\varphi_2\ ?\ \Phi_1 : \Phi_2) : v_0$.

We now proceed to describe, for each of the conditions 1-6, a number of equations that ensure that this condition holds. By an equation, we mean something of the form $\Phi = \Phi'$, where Φ and Φ' are core-wFO formulas. Notice that by Lemma 14, any first-order formula can be turned into an equation (as long as we have at least three distinct weights), so for some conditions we give a first-order formula rather than an equation.

A number of equations $\Phi_i = \Phi_i'$ ensures that the condition holds in the sense that for any (w, σ), $[\![\Phi_i]\!](w, \sigma) = [\![\Phi_i']\!](w, \sigma)$ for each i if and only if (w, σ) satisfies the condition. By Lemma 24, once we have a number of equations $\Phi_i = \Phi_i'$ that together ensure that all conditions are satisfied, the equation $\sum_i \Phi_i = \sum_i \Phi_i'$ ensures that all conditions are satisfied, so that (w, σ) satisfies the conditions if and only if $[\![\sum_i \Phi_i]\!](w, \sigma) = [\![\sum_i \Phi_i']\!](w, \sigma)$.

1. We describe this condition using a first order formula and two equations. The formula makes sure that the word is of the form $d_0 d_1 \cdots d_k$, where each d_i is of the form $\texttt{1} y_0 \texttt{1}^{n_1} \texttt{m} \texttt{1}^{n_2} y_1 \cdots \texttt{1}^{n_{K-1}} \texttt{m} \texttt{1}^{n_K} y_K$, where $n_1 \cdots n_K$ is a sequence of non-negative integers and $y_K = \triangleleft$:

$$\begin{aligned}
&(\forall x.\ \neg\texttt{first-cf}(x) \vee (\texttt{1}(x) \wedge \exists y.\ \texttt{nxt}(x,y) \wedge \texttt{smbl}(y))) \wedge (\exists x.\ \texttt{last}(x) \wedge \triangleleft(x)) \\
&\quad \wedge \forall x.\ \neg\texttt{smbl}(x) \vee \texttt{last}(x) \vee \exists y, z.\ \texttt{nxt-sm}(x,z) \wedge x \leq y \leq z \wedge \texttt{m}(y) \\
&\qquad \wedge \forall i.\ i \leq x \vee z \leq i \vee i = y \vee \texttt{1}(i).
\end{aligned}$$

The following equation ensures that the same number of 1's appear before and after m:

$$\sum_x \texttt{m}(x) ? \sum_y \exists x', y'.\ x' \le y \le x \le y' \wedge \texttt{nxt-sm}(x', y') \wedge \texttt{1}(y) ? v_1(x) =$$
$$\sum_x \texttt{m}(x) ? \sum_y \exists x', y'.\ x' \le x \le y \le y' \wedge \texttt{nxt-sm}(x', y') \wedge \texttt{1}(y) ? v_1(x)$$

Finally, in the context of the formula and equation above, the following equation ensures that for every $1 \le i \le K$, $n_i = 2^i$:

$$\sum_x \texttt{smbl}(x) \wedge \neg\texttt{last}(x) ? (\forall y.\ \neg\texttt{smbl}(y) \vee x \le y ? v_1(x)) :$$
$$(\sum_y \exists z.\ \texttt{smbl}(z) \wedge \texttt{nxt-sm}(z, x) \wedge z \le y \le x \wedge \texttt{1}(y) ? v_1(x)) \quad =$$
$$\sum_x \texttt{smbl}(x) \wedge \neg\texttt{last}(x) ? \sum_y \exists z.\ \texttt{m}(z) \wedge \texttt{nxt-sm}(x, z) \wedge x \le y \le z \wedge \texttt{1}(y) ? v_1(x)$$

2. For this condition, it suffices to require that between each pair of state symbols, there is a $\triangleleft$ symbol, and between two occurrences of $\triangleleft$, there is a state symbol, and right after each state symbol, there is a symbol from the alphabet. The following first-order formula expresses this:

$$\forall x, y.\ \neg\triangleleft(x) \vee \neg\triangleleft(y) \vee \neg x \le y \vee \exists z.\ x \le z \le y \wedge \bigvee_{q \in Q} q(z)$$
$$\wedge\ \forall x, y.\ \neg \bigvee_{q \in Q} q(x) \vee \neg \bigvee_{q \in Q} q(y) \vee \neg x \le y \vee \exists z.\ x \le z \le y \wedge \triangleleft(z)$$
$$\wedge\ \forall x.\ \neg \bigvee_{q \in Q} q(x) \vee \exists y.\ \texttt{smbl}(y) \wedge \texttt{nxt-sm}(x, y) \wedge \neg\triangleleft(y)$$

3. This condition can be imposed by a first order formula that explicitly describes c_0.

4. By the first-order formula $\exists x.\ H(x) \wedge \forall y.\ \neg\triangleleft(y) \vee \neg y \ge x \vee \texttt{last}(y)$.

5. We demonstrate how to treat case a. The other cases are analogous. Fix a transition $(q, s, q's', L) \in \delta$ and $d \in \Sigma$. We use the following shorthand.

$$\texttt{tr}(x, y, z) \overset{\mathsf{def}}{=} q(y) \wedge y \le x \wedge \forall y'.\ \neg(\triangleleft(y') \wedge y \le y' \le x) \wedge s(z) \wedge \texttt{nxt-sm}(y, z) \quad \text{and}$$

$$\texttt{tr'}(x, y, z) \overset{\mathsf{def}}{=} q'(y) \wedge y \le x \wedge \forall y'.\ \neg(\triangleleft(y') \wedge y \le y' \le x) \wedge s'(z) \wedge \texttt{nxt-sm}(y, z)$$

Let $s_1, s_2, \ldots, s_m$ be a permutation of Σ. We use the following equation:

$$\sum_x \triangleleft(x) \wedge \exists y.\ (\triangleleft(y) \wedge \neg y \leq x) \wedge \exists y, z.\ \mathtt{tr}(x,y,z)?$$
$$\sum_y \mathtt{smbl}(y) \wedge y \leq x \wedge \forall z.\ (x \leq z \vee \neg y \leq z \vee \neg\triangleleft(x))?$$
$$q(y)\ ?\ \mathtt{ps}_{v_q^x}(y) : s_1(y)\ ?\ \mathtt{ps}_{v_{s_1}^x}(y) : s_2(y)\ ?\ \mathtt{ps}_{v_{s_2}^x}(y) : \cdots : s_m(y)\ ?\ \mathtt{ps}_{v_{s_m}^x}(y)$$
$$=$$
$$\sum_x \triangleleft(x) \wedge \exists y.\ (\triangleleft(y) \wedge \neg y \leq x) \wedge \exists y, z.\ \mathtt{tr}(x,y,z)?$$
$$\sum_y \mathtt{smbl}(y) \wedge x \leq y \wedge \forall z.\ (z \leq x \vee \neg z \leq y \vee \neg\triangleleft(x))?$$
$$q'(y) \wedge \exists z.\ \mathtt{smbl}(z) \wedge \mathtt{nxt\text{-}sm}(y,z) \wedge s_1(z)\ ?\ \mathtt{ps}_{v_{s_1}^x}(y) :$$
$$q'(y) \wedge \exists z.\ \mathtt{smbl}(z) \wedge \mathtt{nxt\text{-}sm}(y,z) \wedge s_2(z)\ ?\ \mathtt{ps}_{v_{s_2}^x}(y) : \cdots$$
$$q'(y) \wedge \exists z.\ \mathtt{smbl}(z) \wedge \mathtt{nxt\text{-}sm}(y,z) \wedge s_m(z)\ ?\ \mathtt{ps}_{v_{s_m}^x}(y) :$$
$$\exists z.\ q'(z) \wedge \mathtt{nxt\text{-}sm}(z,y)\ ?\ \mathtt{ps}_{v_q^x}(y) :$$
$$\exists z, z'.\ q'(z) \wedge \mathtt{nxt\text{-}sm}(z,z') \wedge \mathtt{nxt\text{-}sm}(z',y)\ ?\ \mathtt{ps}_{v_s^x}(y) :$$
$$s_1(y)\ ?\ \mathtt{ps}_{v_{s_1}^x}(y) : s_2(y)\ ?\ \mathtt{ps}_{v_{s_2}^x}(y) : \cdots : s_m(y)\ ?\ \mathtt{ps}_{v_{s_m}^x}(y)$$

The rightmost part of the equation ensures that if the effects of the transition are reversed, then all symbols are in the same place as in the previous configuration. We can then make sure that the state has changed to q' and the symbol to s' with the following formula:

$$\forall x, y.\ \neg(\triangleleft(x) \wedge \triangleleft(y) \wedge \neg y \leq x \wedge \exists x_q, x_s.\ \mathtt{tr}(x, x_q, x_s)) \vee \exists y_q, y_s.\ \mathtt{tr'}(y, y_q, y_s).$$

Proof of Theorem 25. We use a reduction from the Halting Problem, as it is described above. It is not hard to see why conditions 1 to 5 suffice for the correctness of the reduction, and it is not hard to see that the formulas we construct ensure the corresponding conditions. Furthermore, notice that all formulas are core-wFO formulas, and therefore the problem is undecidable for core-wFO, but also for core-wMSO, which is a more general case. ◀