On the complexity of perfect models of logic programs

Sergey M. Dudakov
Dept. of CS, Tver State Univ.
33 Zhelyabova str., Tver, 170013, Russia
p000104@tversu.ru

Abstract. In this paper we investigate computational complexity of the PERF-consistency and PERF-entailment problems for ground normal logic programs. In [3] it is proved that these problems belong to $\Sigma_2^p$ and $\Pi_2^p$ correspondingly. The question of obtaining more accurate results was left as open. We prove that both problems belong to $\Delta_2^p$. Lower bounds on the complexity of these problems are also established in terms of a new complexity class $D_2$ which is a subset of $\Delta_2^p$. It is shown that PERF-consistency is a $D_2$-complete problem and PERF-entailment is co-$D_2$-complete.

1. Introduction

A number of proposals are well-known for intended semantics of logic programs (LP) which have negative subgoals (normal programs) (see, e.g., [1]). One of them is the perfect model semantics which was introduced in [7]. It was shown that if a logic program is stratified [2] then it has a perfect model, and this model is its unique stable model [4]. Moreover, when some logic program has a perfect model, then this perfect model is the unique [7, 8]. The complexity problems for stable models were investigated in [5] where it was proved that the problem of the existence of a stable model for propositional programs is NP-complete (see also [6] for more detailed treatment). There was less information on the complexity of the perfect model semantics. In [3] two algorithmic problems were considered for propositional disjunctive logic programs: given LP $P$, decide whether $P$ has a perfect model (PERF-consistency), and given LP $P$ and propositional formula $\Phi$, decide whether any perfect model of $P$ satisfies $\Phi$ (PERF-entailment). It was proved that these problems are $\Sigma_2^p$-complete and $\Pi_2^p$-complete correspondingly. For nondisjunctive LPs it was shown that both problems belong to the corresponding classes and that the set UMINSAT (a set of CNF having exactly one minimal model) is m-reducible to them.

In this paper we prove that both problems belong to $\Delta_2^p$, i.e. they are computable on deterministic Turing machines with NP-complete oracle in polynomial time. This result gives a
negative answer to the following questions from [3]: “Is PERF-consistency $\Sigma_2^P$-complete?” and “Is PERF-entailment $\Pi_2^P$-complete?”

We investigate also the lower bounds on the complexity of these problems. A new kind of Turing machines with oracles is defined. The machines have an oracle with a fixed subset of forbidden elements, and a question on any such element immediately causes the machine to reject its input. $D_2$ is the class of sets that can be recognized by such machines in polynomial time with oracle SAT when the set of “forbidden” questions is SAT \ UMINSAT (a set of CNF which have more than one minimal model). We prove that PERF-consistency is $D_2$-complete and PERF-entailment is co-$D_2$-complete.

2. Notions and definitions

Let $A$ be a set of ground atoms. We consider logic programs over $A$. A literal is an atom or a negation of atom. Logic program is a finite set of clauses of the form $Head ← Body$, where $Head$ is an atom and $Body$ is a conjunction of literals. Usually such LPs are called normal LPs. An interpretation is any subset of $A$. For $a ∈ A$, $I |{}= a$ if $a ∈ I$, and $I |{}= ¬a$ if $a /∈ I$. $I |{}= h ← body$ if either $I |{}= h$ or there is a literal $l ∈ body$ such that $I /{}= l$. Let $P$ be an LP; $I$ is a model of $P (I |{}= P)$ if for all clauses $γ$ of $P I |{}= γ$. Given LP $P$, only some finite subset $A_P ⊆ A$ which includes all atoms of clauses of $P$ is relevant for the semantics of $P$. We assume that for every $P$ such $A_P$ is fixed. Below the index $P$ will therefore omitted.

Let $\text{Cnf}(P)$ denote a boolean formula in conjunctive normal form (CNF) which is equivalent to $P$.

We define a perfect model as in [3]. To do this we need some auxiliary ordering relations on $A$.

**Definition 2.1.** Let LP $P$ contain a clause $a ← b_1, ..., b_m, ¬c_1, ..., ¬c_n$. Then $a ≤_1 b_j$ and $a ≤_1 c_k$ for all $j = 1 \ldots m$ and $k = 1 \ldots n$. $≤$ and $≤$ are the smallest extensions of $≤_1$ and $≤_1$ such that for any $a, b, c ∈ A$:

1) $a ≤ b$ $⇒$ $a ≤ b$;
2) $a ≤ b$ and $b ≤ c$ $⇒$ $a ≤ c$;
3) $a ≤ b$ and $b ≤ c$ ($c ≤ a$) $⇒$ $a ≤ c$ ($c ≤ b$).

We write $a ≡ b$ if $a ≤ b$ and $b ≤ a$. By $\text{Equ}(a)$ we denote the equivalency class of $a$. $a < b$ ($a < b$) if $a ≤ b$ ($a ≤ b$) and $a ≢ b$.

Now we define a perfect model of $P$.

**Definition 2.2.** Let $I_1$ and $I_2$ be two models of LP $P$. $I_1 ≪ I_2$ if for all $x ∈ I_2 \setminus I_1$ there exists $y ∈ I_2 \setminus I_1$ such that $x ≤ y$. A model $I$ of LP $P$ is perfect if there is no model $I'$ such that $I' ≪ I$.

Let us consider some examples.

**Example 2.1.** The classic example of a logic program which has no perfect model is the following program:

$$P = \{a ← ¬b, b ← ¬a\}.$$
This LP has three models $I_1 = \{a\}$, $I_2 = \{b\}$, and $I_3 = \{a, b\}$. Here, $a \leq b$ and $b \leq a$, so by the definition $I_1 \ll I_2$, $I_2 \ll I_1$, $I_1 \ll I_3$ and $I_2 \ll I_3$. We see that no model satisfies the definition of a perfect model.

**Example 2.2.** Let us consider another LP

$$P_1 = \{a \leftarrow \neg b, b \leftarrow \neg a, a \leftarrow \neg a\}.$$ This LP has two models $I_1 = \{a\}$ and $I_2 = \{a, b\}$. By the definition $I_1 \ll I_2$, hence, $I_1$ is the perfect model of $P_1$.

The following two notions are necessary for more technical reasons and are used in many proofs below.

**Definition 2.3.** Let $I_1$ and $I_2$ be two interpretations, $a \in A$. $I_1 \equiv \succeq a I_2$ ($I_1 \equiv \succ a I_2$) iff for any atom $x \succeq a$ ($x \succ a$) $x \in I_1 \iff x \in I_2$.

**Definition 2.4.** Let $I$ be an interpretation, $d \in A$. Then set

$$S^I(d) = \{x \in \text{Equ}(d) : \exists I' \equiv \succ_d I \text{ such that } x \notin I' \text{ and } I' \models P\},$$
i.e. $S^I(d)$ is a set of atoms $x$ which are equivalent to $d$ and for which there exists a model $I'$ of $P$ that is equal to $I$ on atoms greater than $d$, and $I' \models \neg x$.

We will also use well-known closer operator for the inflationary semantics.

**Definition 2.5.** Let $S \subseteq A$ and $P$ be LP on $A$. Then we set

$$\text{Cl}^P_1(S) = S \cup \{x : x \leftarrow \alpha \in P \text{ and } S \models \alpha\}; \text{ and } \text{Cl}^P(S) = \bigcup_{i=0}^{\infty} (\text{Cl}^P_1)^i(S).$$

So $\text{Cl}^P(S)$ is the set of all atoms which can be derived from $S$. It is clear that for all $S$ $\text{Cl}^P(S) \models P$.

Let SAT be the set of all CNFs which have a model and UMINSAT be the set of all CNFs having an unique minimal (with respect to $\subseteq$) model.

We investigate the complexity of two following problems:

1. **PERF-consistency**: Given LP $P$, decide if $P$ has a perfect model.
2. **PERF-entailment**: Given LP $P$ and propositional formula $\Phi$, decide whether any perfect model $I$ of $P$ satisfies $\Phi$.

### 3. Upper bound

In this section we present an algorithm that constructs a perfect model of given LP or decides that such model doesn’t exist. Therefore, it solves PERF-consistency problem. The algorithm uses NP-complete oracle SAT and works in polynomial time. This proves that PERF-consistency belongs to $\Delta_2^P$. 
Algorithm PERFCONS
Input: a set of atoms $A$ and LP $P$.
Output: a perfect model of $P$ or “no” if it doesn’t exist.

begin
0. Compute the relation $\preceq$ for $P$.
1. $I_0 = A$; $Done = \emptyset$; $i = 1$;
2. do
3. Find any maximal (with respect to $\preceq$) atom $a$ in $I_{i-1} \setminus Done$;
4. $S' = \emptyset$;
5. for all $x \in Equ(a)$
6. Ask if “Cnf($P$) $\land \bigwedge_{y \in_{i-1} a} y \land \bigwedge_{y \notin_{i-1} a} \neg y \land \neg x \in SAT?”.
7. if the answer is “yes” then $S' = S' \cup \{x\}$ endif
8. endfor;
9. Ask if “Cnf($P$) $\land \bigwedge_{y \in_{i-1} S'} y \land \bigwedge_{y \notin_{i-1} S'} \neg y \in SAT?”.
10. if the answer is “no” then return “no” endif;
11. $I_i = I_{i-1} \setminus S'$; $Done = Done \cup Equ(a)$; $i = i + 1$
12. while $Done \neq A$
13. return $I_i$
end

The question in line 6 is to check whether there is $I'$ such that $I' \equiv_{\geq a} I_{i-1}$, $x \notin I'$, and $I' \models P$; the question in line 9 is to check whether there is $I'$ such that $I' \equiv_{\geq a} I_{i-1}$, $I' \cap S' = \emptyset$ and $I' \models P$. The relation $\preceq$ is computable in polynomial time. The loop 2–12 is repeated no more than $|A|$ times, the search in line 3 can be done in $O(|A|)$ steps, the loop 5–8 is repeated no more than $|A|$ times. All other lines require constant time. Hence, the algorithm PERFCONS computes its result in polynomial time.

To prove that PERFCONS is correct, we need the following lemma.

Lemma 3.1. Given LP $P$ and an interpretation $I$, let $d \in I$ be such that
1) for any $x \in I$ if $x \succ d$ then $S^I(x) \cap I = \emptyset$ and
2) $S^I(d) \cap I \neq \emptyset$.
Let $I_1$ be any perfect model of $P$ such that $I_1 \equiv_{\geq d} I$. Then $I_1 \cap S^I(d) = \emptyset$.

Proof:
Let $I_1 \equiv_{\geq d} I$ be any perfect model of $P$. Let us suppose that $I_1 \cap S^I(d) \neq \emptyset$. We disprove this supposition by constructing a model $I_3$ of $P$ such that $I_3 \preceq I_1$. Let $a \in I_1 \cap S^I(d)$.

From the definition of $\preceq$ it follows that two cases are possible:
(i) for all $x, y \in Equ(d)$ $x \preceq y$ or
(ii) for all $x, y \in Equ(d)$ $x \not\preceq y$.
In case (i) we fix $I_2 \equiv_{\geq d} I_1$ such that $a \notin I_2$ and $I_2 \models P$. Such $I_2$ exists, since $a \in S^I(d)$. Let $I' = (I_1 \setminus Equ(d)) \cup (I_2 \cap Equ(d))$ and $I_3 = Cl^P(I')$. In the case (ii) we define $I' = I_1 \setminus S^I(d)$.
and $I_3 = \text{CI}^P(I')$. It is easy to check that in both cases $a \not\in I_3$ and if $x \in I_3 \setminus I_1$ then $x \preceq a$. So $I_3 \ll I_1$ and $I_1$ is not a perfect model, which contradicts to the condition of the lemma. \hfill \square

Lemma 3.2. The algorithm PERFCONS is correct.

Proof:
Let $J$ be a perfect model of $P$. We prove that for any $i$ after $i$ executions of the loop (2)–(12) for all $x \in \text{Done}$, the following condition is satisfied: $J \equiv_{\preceq, x} I_i$ and $I_i \cap S^I_i(x) = \emptyset$. It holds trivially before the start of the loop. Suppose, it holds before the $i$-th execution. Let an atom $a$ be selected in line 3. It is enough to prove that $i$ is trivially before the start of the loop. Suppose, it holds before the $i$-th execution. Let an atom $a$ be selected in line 3. It is enough to prove that $J \equiv_{\preceq, a} I_i$ and $I_i \cap S^I_i(a) = \emptyset$. Let $x \equiv a$. If $x \not\in J$, then $x \in S'$ and $x \not\in I_i$. If $x \not\in I_i$, then $x \in S'$. Note that $S' = S^{I_i-1}(a) = S^I_i(a)$. Thus, $S^{I_i-1}(a) \cap I_{i-1} \neq \emptyset$. By lemma 3.1 we have that $J \cap S' = \emptyset$. Hence, $x \not\in J$ and $I_i \cap S^I_i(a) = \emptyset$. After the last $n$-th execution we have $\text{Done} = A$ and $I_n = J$.

Let PERFCONS return $J$. Let $\gamma = a \leftarrow \text{body}$ be any clause of $P$. Suppose that $\text{Equ}(a)$ is considered at the $i$-th execution. Since the oracle’s answer to the question in line (9) was “yes”, there is a model $I'$ of $P$ such that $I' \equiv_{\preceq, a} I_i$ and $I' \cap S' = \emptyset$. It means that $I' \cap \text{Equ}(a) \subseteq I_i \cap \text{Equ}(a)$. If $x \in I_i \cap \text{Equ}(a)$, then $x \not\in S'$. So $x \in I'$ by the construction of $S'$. We have that $I' \cap \text{Equ}(a) = I_i \cap \text{Equ}(a)$. Hence, $I' \equiv_{\preceq, a} I_i$ and $I_i = \gamma$. Since $J \equiv_{\preceq, a} I_i$, $J = \gamma$.

Suppose that $P$ has a model $J' \ll J$. Let $a_0$ be any maximal (with respect to $\preceq$) atom in $J \setminus J'$. If $J \not\equiv_{\preceq, a_0} J'$, then there exists an atom $a' > a_0$ such that $a' \in J \setminus J'$ or $a' \in J' \setminus J$. In the first case $a_0$ is not maximal, the second contradicts to $J' \ll J$. So $J \equiv_{\preceq, a_0} J'$. At the step where $a_0$ is considered, $a_0 \in S'$, hence, $a_0 \not\in J$, which contradicts to $a_0 \in J \setminus J'$. Therefore, $J$ is a perfect model of $P$. \hfill \square

From lemma 3.2 and the time bound on PERFCONS, we get the following theorem.

Theorem 3.1. PERF-consistency belongs to $\Delta^P_2$.

Corollary 3.1. PERF-entailment belongs to $\Delta^P_2$.

Proof:
To prove the corollary, we describe an algorithm solving PERF-entailment. Given LP $P$ and propositional formula $\Phi$, run PERFCONS for $P$, if it returns “no” then the answer is “yes”. If it returns $I$, check if $I \models \Phi$. Since the perfect model is unique, the algorithm is correct. \hfill \square

4. Lower bound
To establish the lower bound of the complexity of the problems under consideration, we define a new complexity class. Let us describe some kind of deterministic Turing machines with oracle.

Let $M^P = (Q, \Sigma, Pr, q_0, q_a, q_r)$, where $Q$ is a set of states, $\Sigma$ is a tape alphabet, $Pr$ is a program, $q_0, q_a, q_r \in Q$ are the initial, accept and reject states respectively. We also distinguish states $q_2, q_0, q_a, q_r \in Q$ as, respectively, the query state and two answer states. An oracle is a pair of sets $(A, B)$ such that $A \cap B = \emptyset$. If the current state of the machine is $q_2$ and $x$ is a query, then at
the next time the state is $q_x$ if $x \in A$, $q_-$ if $x \in B$, and $q_r$ otherwise. Therefore $\Sigma^* \setminus (A \cup B)$ is the set of "forbidden" questions.

Let $P^{(A,B)}$ be the class of sets which can be recognized by machines $M^{(A,B)}$ in polynomial time. Clearly, $\text{SAT} \cap \text{UMINSAT} = \emptyset$, so we can use the pair ($\text{UMINSAT}$, $\text{SAT}$) as an oracle.

**Definition 4.1.** Set $D_2 = P(\text{UMINSAT}, \text{SAT})$.

Since $\text{SAT} \in \Delta^P_2$ and $\text{UMINSAT} \in \Delta^P_2$ so $D_2 \subseteq \Delta^P_2$.

We prove that PERF-consistency is a $D_2$-complete problem and PERF-entailment is co-$D_2$-complete.

**Lemma 4.1.** PERF-consistency belongs to $D_2$.

**Proof:**
To prove the lemma, we construct the algorithm PERFCONS1. It is obtained from PERFCONS by deleting lines 6, 9–10 and adding lines 4.1–4.3 and 6.1:

1. $P' = \{ x \leftarrow b \in P : x \in \text{Equ}(a) \}$.
2. Replace all atoms $y > a$ in $P'$ by its values in $I_{i-1}$.
3. Ask the oracle about $\varphi = \text{Cnf}(P')$.

6.1. Ask the oracle about $\psi = \varphi \land \neg x$.

The oracle is ($\text{UMINSAT}$, $\text{SAT}$). If PERFCONS1 asks about elements of ($\text{SAT} \setminus \text{UMINSAT}$), then it immediately stops and returns "no". We prove that the algorithms PERFCONS and PERFCONS1 are equivalent. It is enough to prove that lines 4.1–4.3 of PERFCONS1 do the same work as lines 9–10 of PERFCONS and line 6.1 is equivalent to 6.

Suppose that line 4.3 rejects $P$. So $\varphi$ has at least two minimal models $J_1$ and $J_2$. Let us consider the work of PERFCONS. Clearly, $S' \supseteq (\text{Equ}(a) \setminus J_1) \cup (\text{Equ}(a) \setminus J_2)$. If $P$ has a model $I' \equiv_a I_{i-1}$ such that $I' \cap S' = \emptyset$, then $J = I' \setminus \text{Equ}(a)$ is a model of $\varphi$. It contradicts the minimality of $J_1$ and $J_2$. Consequently, PERFCONS also returns "no". Let $\varphi$ have a unique minimal model $J$. Then $S' = \text{Equ}(a) \setminus J$, and $I' = I_{i-1} \setminus S'$ is a model of $P$. The case $\varphi \notin \text{SAT}$ is impossible because each disjunction of $\varphi$ has at least one positive literal, hence, $\text{Equ}(a)$ is a model of $\varphi$. We proved that in line 4.3 PERFCONS1 continues iff the condition in line 10 of PERFCONS is false. Hence, line 4.3 is equivalent to 9–10.

Now let us prove that in line 6.1 PERFCONS1 never asks questions about $\psi \in \text{SAT} \setminus \text{UMINSAT}$. Let $\psi$ have two minimal models $J_1$ and $J_2$. Since $\psi = \varphi \land \neg x$ and $\varphi$ has an unique minimal model $J$ (line 4.3), then $J_1 \models \varphi$ and $J_2 \models \varphi$. Hence, $J \subseteq J_1 \cap J_2$, $J \models \neg x$ and $J \models \psi$, which contradicts the minimality $J_1$ and $J_2$. \hfill \Box

Now let us establish some useful properties of CNFs.

**Lemma 4.2.** Let $\text{CNF } \Phi$ consist of $m$ disjunctions. Then for any $n \geq m$, a $\text{CNF } \Psi$ can be constructed in polynomial time, such that

1. $\Psi \in \text{SAT} \iff \Phi \in \text{SAT}; \Psi \in \text{UMINSAT} \iff \Phi \in \text{UMINSAT}$;
2. $\Psi$ consists of $n$ disjunctions, and $\Psi$ can be constructed in polynomial time by $\Phi$.\hfill \Box
Proof:
Let \( k = n - m \) and let \( y_1, \ldots, y_k \) be new atoms. Let \( \Psi = \Phi \land \neg y_1 \land \ldots \land \neg y_k \). Evidently, \( \Psi \) satisfies the conditions of the lemma. \( \square \)

Lemma 4.3. Let \( \Phi \) be a CNF. Then a CNF \( \Psi \) can be constructed in polynomial time, such that:
1. \( \Psi \in SAT \iff \Phi \in SAT; \ \Psi \in UMINSAT \iff \Phi \in UMINSAT; \)
2. each disjunction of \( \Psi \) contains no more than two negative literals.

Proof:
If \( \Phi \) doesn’t satisfy 2, then \( \Phi = (\neg a \lor \neg b \lor D) \land \Phi' \) where \( D \) contains negations. Let \( c \) be a new atom and \( \Phi^* = (\neg a \lor \neg b \lor c) \land (\neg c \lor D) \land \Phi' \). It is easy to show that \( \Phi \in SAT \iff \Phi^* \in SAT \) and \( \Phi \in UMINSAT \iff \Phi^* \in UMINSAT \). Now the lemma follows by standard induction. \( \square \)

Lemma 4.4. PERF-consistency is a \( D_2 \)-hard problem.

Proof:
Let \( A \in D_2 \). We will show how to reduce the question "\( x \in A \)" to the PERF-consistency problem for some LP \( P \), which can be constructed in polynomial time.

Since \( A \in D_2 \), there is a machine \( M^{(UMINSAT,SAT)} \) recognizing \( A \) in polynomial time. Let \( M^{(1)} = (Q, \Sigma, Pr, q_0, q_a, q_r) \), and let \( Q \) contain \( q_a, q_r, q_+, q_- \). Let \( q_a, q_r, q_? \) occur only in the right parts of the instructions, and let \( q_+, q_- \) occur only in their left parts. Let \( M^{(1)} \) work on input \( x \) in time \( T = pol(|x|) \). For convenience, we suppose that the time of \( M \) is exactly \( T \) when no "forbidden" question is asked. We also suppose that \( M^{(1)} \) is right-sided, and that the left part of the tape is used for the queries. We will also suppose that all queries contain the same number of disjunctions \( K \), and each disjunction contains no more than two negations. Lemmas 4.2 and 4.3 show that this is possible without loss of generality. Evidently, the number \( L \) of variables in each query is bounded by some polynomial of the input size. So we suppose that in all queries the variables are from the list \( y_1, \ldots, y_L \).

Now we describe the standard form of a query. Let \( M^{(k)} \) write queries in cells \(-1, -2, \ldots, -KL\). This part of the tape is divided into \( K \) sections and each of them consists of \( L \) cells. The number of the \( l \)-th cell in the \( k \)-th section is \((k, l) = -(k-1)*L + l\). Let \( \Sigma \) include 3 special symbols: \(+\), \(-\), and \(0\). When the machine asks the oracle, i.e. turns into \( q_? \), the content of the left side of the tape encodes a CNF in question as follows: cell \((k, l)\) contains \(+\) if \( y_l \) belongs to the \( k \)-th disjunction, it contains \(-\) if \( \neg y_l \) belongs to the \( k \)-th disjunction, and 0 if neither \( y_l \) nor \( \neg y_l \) belong to the \( k \)-th disjunction.

Now we describe the set of atoms \( A \) needed to construct \( L \). Below we use variables with indexes, for example \( h_{l,i} \). We suppose that an index \( t \) is in \([0, T]\) (a moment of time), \( i, i_1, i_2 \) in \([-KL, T]\) (the tape space), \( k \) in \([1, K]\) (number of a disjunction), \( l, l_1, l_2 \) in \([1, L]\) (numbers of variables). \( A \) consists of the following sets of atoms:

1. \( q_{t,s} \), for all \( s \in Q \). \( q_{t,s} \) denotes that at moment \( t \) \( M \) is in state \( s \).
2. \( h_{t,i} \) means that at moment \( t \) the head of \( M \) is in cell \( i \).
3. \(a_{t,i,b}\), for all \(b \in \Sigma\). \(a_{t,i,b}\) means that at moment \(t\) cell \(i\) contains \(b\).
4. \(y_{t,j}\), \(y_{t,l,k}\) are used to model the oracle answers. They describe the minimal model of a query if it exists.
5. \(w', w''\) prevent \(L\) from having a perfect model in the case of rejection.
6. \(a\) is used to construct the relation \(\leq\) on atoms.

For every instruction \(sb \rightarrow s'b'c\) from \(Pr\) where \(s, s' \in Q, b, b' \in \Sigma, c \in \{-1,0,+1\}\) we include the following clauses in \(P\):

\[
\begin{align*}
q_{t+1,s'} &\leftarrow q_{t,s}, h_{t,i}, a_{t,i,b}; \quad h_{t+1,i,c} &\leftarrow q_{t,s}, h_{t,i}, a_{t,i,b} \\
q_{t+1,i,b'} &\leftarrow q_{t,s}, h_{t,i}, a_{t,i,b}
\end{align*}
\] (1)

for all \(t\) and \(i\).

The clauses (2) determine when the configuration parameters do not change.

\[
\begin{align*}
a_{t+1,i,b} &\leftarrow \neg h_{t,i}, a_{t,i,b}; \quad h_{t+1,i} &\leftarrow q_{t,q_r}, h_{t,i}; \quad a_{t+1,i,b} &\leftarrow q_{t,q_r}, h_{t,i}, a_{t,i,b};
\end{align*}
\] (2)

for all \(b \in \Sigma, t\) and \(i\).

The clauses (3) say that at every moment there is a unique state, a unique head position, and each cell contains a unique symbol.

\[
\begin{align*}
q_{t+1,q_r} &\leftarrow h_{t,i_1}, h_{t,i_2}; \quad q_{t+1,q_r} &\leftarrow q_{t,q_1}, q_{t,q_2}; \quad q_{t+1,q_r} &\leftarrow a_{t,i,b_1}, a_{t,i,b_2};
\end{align*}
\] (3)

for all \(b_1, b_2 \in \Sigma, b_1 \neq b_2, q_1, q_2 \in Q, q_1 \neq q_2, i_1 \neq i_2\) and \(t\).

Clauses (4)–(7) are used to simulate the answers of the oracle.

\[
\begin{align*}
y_{t+1,l,k} &\leftarrow q_{t,q_r}, y_{t+1,l}, \neg a_{t,(k,l),0}; \quad y_{t+1,l} &\leftarrow q_{t,q_r}, y_{t+1,l,k}, \neg a_{t,(k,l),0} \\
y_{t+1,q_r} &\leftarrow q_{t,q_r}, y_{t+1,l,k}, a_{t,(k,l)}
\end{align*}
\] (4)

These clauses mean that if \(\varphi\) is a query and \(y_l\) occurs in \(k\)-th disjunction of \(\varphi\) then \(y_{t+1,l}\) is equivalent to \(y_{t+1,l,k}\). Otherwise, \(y_{t+1,l,k}\) should be false.

\[
\begin{align*}
q_{t+1,q_{-}} &\leftarrow q_{t,q_r}, \bigwedge_{l \in [1,L]} \neg y_{t+1,l,k}, \neg a_{t,(k,l),-} \\
q_{t+1,q_{+}} &\leftarrow q_{t,q_r}, y_{t+1,l,k}, a_{t,(k,l),-}, y_{t+1,l,k}, a_{t,(k,l),-}, \bigwedge_{l \in [1,L]} \neg y_{t+1,l,k}, \neg a_{t,(k,l),-}
\end{align*}
\] (5)

for all \(t, k, l_1 \leq l_2\). Note that if \(\varphi\) is not satisfied by the interpretation described by \(y_{t+1,l}\)’s, then the body of one of these clauses has to be true.

\[
\begin{align*}
y_{t+1,l} &\leftarrow q_{t,q_r}, q_{t+1,q_{-}} \\
q_{t+1,q_{+}} &\leftarrow q_{t,q_r}, \neg q_{t+1,q_{-}}
\end{align*}
\] (6) (7)

for all \(t\). The initial state is described by the clauses:

\[
\begin{align*}
h_{0,0} &\leftarrow; \quad q_{0,q_0} &\leftarrow; \quad a_{0,i,b} &\leftarrow,
\end{align*}
\] (8)
if at the initial moment cell \( i \) contains symbol \( b_i \).

The next two clauses provide that when an input is rejected then \( P \) does not have a perfect model.

\[
\begin{align*}
    w' & \leftarrow q_{t,q_t}, \neg w''; \quad w'' \leftarrow q_{t,q_t}, \neg w'.
\end{align*}
\]  

The last set of clauses defines the ordering of atoms:

\[
\begin{align*}
    v_{t+1} & \leftarrow \neg u_t, a, \neg a; \\
    u_t & \leftarrow \neg v_t, a, \neg a,
\end{align*}
\]  

where \( v_t, u_t, v_{t+1} \) are any atoms having, respectively, \( t \) and \( t + 1 \) as their first indexes, except of \( v_t = q_{t,q_t} \). Therefore, \( v_{t+1} < u_t, q_{t,q_t} < v_t \), and \( u_t \equiv v_t \) if \( v_t \neq q_{t,q_t} \).

Now it can be proved (by a long but routine argumentation) that \( P \) has a perfect model iff \( M \) accepts \( x \), which finishes the proof of the lemma. \( \square \)

From lemmas 4.1 and 4.4, follows

**Theorem 4.1.** PERF-consistency is a D\(_2\)-complete problem.

Let us consider the complexity of PERF-entailment. In [3] it is proved that the negation of PERF-consistency is m-reducible to PERF-entailment. It gives the lower bound in the following

**Theorem 4.2.** PERF-entailment is co-D\(_2\)-complete problem.

**Proof:**

The upper bound is also easy to establish. To check if LP \( P \) has a perfect model \( I \) such that \( I \models \Phi \), it is enough to do the following: use PERFCONS1 to construct a perfect model \( I \), if it doesn’t exist then an answer is “no”. Otherwise check whether \( I \models \neg \Phi \). \( \square \)

**References**


