Nov. 2005, Vol.20, No.6, pp.758–762 *J. Comput. Sci. & Technol.*

Complexities of Homomorphism and Isomorphism for Definite Logic Programs

Dao-Yun Xu[1] (许道云) and Zhi-Hong Tao[2] (陶志红)

[1]*Department of Computer Science, Guizhou University, Guiyang 550025, P.R. China*
[2]*Department of Computer Science, Peking University, Beijing 100871, P.R. China*

E-mail: dyxu@gzu.edu.cn

Received December 27, 2004; revised April 8, 2005.

Abstract A homomorphism φ of logic programs from P to P' is a function mapping $Atoms(P)$ to $Atoms(P')$ and it preserves complements and program clause. For each definite program clause $a \leftarrow a_1, \ldots, a_n \in P$ it implies that $\varphi(a) \leftarrow \varphi(a_1), \ldots, \varphi(a_n)$ is a program clauses of P'. A homomorphism φ is an isomorphism if φ is a bijection. In this paper, the complexity of the decision problems on homomorphism and isomorphism for definite logic programs is studied. It is shown that the homomorphism problem (*HOM-LP*) for definite logic programs is *NP*–complete, and the isomorphism problem (*ISO-LP*) is equivalent to the graph isomorphism problem (*GI*).

Keywords logic program, homomorphism, isomorphism, decision problem, complexity

1 Introduction

A literal is a propositional variable or a negated propositional variable. A clause C can be regarded as a set of literals and a *CNF* formula F as a set of clauses. A clause C is a Horn clause if C contains at most one positive literal. A Horn clause C is definite if C contains exactly one positive literal. A definite logic program is a set of definite Horn clauses. For *CNF* formulas H and F, a homomorphism φ from the formula H to F is a mapping from $lit(H)$ to $lit(F)$ and it preserves complements and clauses, i.e., $\varphi(\neg L) = \neg\varphi(L)$ for $L \in lit(H)$, and $\varphi(C) \in F$ for every clause $C \in H$, where $lit(\cdot)$ is the set of literals over variables occurring in the formula, and φ is an isomorphism from the formula H to F if φ is a homomorphism from the formula H to F and φ is a bijection. Clearly, if the formula H is homomorphic to the formula F, then the unsatisfiability of H implies the unsatisfiability of F, and if the formula H is isomorphic to the formula F, then H and F have the same satisfiability.

We are interested in homomorphism and isomorphism of *CNF* formulas for motivations of constructing some more efficient algorithms for satisfiability and simplifying the proofs of unsatisfiable formulas[1−6]. In [2], Krishnamurthy illustrated the power of symmetry for propositional proof systems. He added to the resolution calculus the rule of symmetry and gave short proofs for some hard formulas. For example, the pigeon hole formulas have a proof of polynomial size in this extended calculus. The rule of symmetry allows the following inference: if a clause C has been derived from a set of clauses F and φ is a permutation over the set of variables occurring in F, then the clause $\varphi(C)$ can be inferred as the next step in the derivation. Further interesting results can be found in Urquhart's paper[6]. We call a permutation of variables as variable renaming. Instead of a permutation of variables we can make use of a more general renaming, namely a so-called literal renaming or isomorphism. That means we have a permutation of variables and additionally variables can be simultaneously replaced by their complements.

A deeper understanding of structures of *CNF* formulas may help to improve DPLL algorithm[7]. In the splitting tree of the DPLL algorithm, if two formulas are labelled at two different nodes, and one of the formulas can be mapped to the other by an isomorphism, then we can replace one of the formulas by the empty clause and continue with the remaining formula. We have showed that DPLL algorithm with such a symmetry rule has short proofs for the pigeon hole formulas, which are a class of hard formulas. It needs only a cubic number of nodes[8]. Szeider[9] introduced homomorphisms of formulas for simplifying the proof of the unsatisfiability of formulas and investigated the *core* of formulas. A homomorphism φ from H to F is called a *retraction* if there exists a homomorphism ψ from F to H such that $\psi \circ \varphi = Id_F$, where Id_F is the identity function over F. In this case we call F as a *retract* of H. A formula F is a *core* if every retract of F is isomorphic to F. The formula F is a *core* of H if F is a core and F is a retract of H. Szeider showed that the cores of formulas are mutually isomorphic and the recognition of cores is *co-NP*-complete[9].

The definite logic programs, a class of special *CNF* formulas, have some good structures. It is easy to prove that for given definite logic programs P and Q, if φ is a homomorphism from P to Q with $\varphi(P) = Q$ and A is the answer set of P, then $\varphi(A)$ is a subset of the answer set of Q, and further, if φ is an isomorphism from P to Q, then $\varphi(A)$ is the answer set of Q. Investigating decision problems on homomorphism and isomorphism

Short Paper
The work is supported by the National Natural Science Foundation of China (Grant Nos. 60310213 and 60463001), the Special Foundation for Improving Scientific Research Condition of Guizhou, and the Government Foundation of Guizhou Province.

for definite logic programs is closely relevant to the construction of some minimal unsatisfiable formulas and the graph isomorphism problem (*GI*). A formula F is minimal unsatisfiable if F is unsatisfiable and $F \setminus \{C\}$ is satisfiable for any clause $C \in F$. In [10], Papadimitriou and Wolfe showed that for every formula F one can construct a formula $f(F)$ in polynomial time such that

- F is satisfiable if and only if $f(F)$ is satisfiable;
- F is unsatisfiable if and only if $f(F)$ is minimal unsatisfiable.

In other words, an unsatisfiable formula can be transformed into a minimal unsatisfiable formula in polynomial time.

The *deficiency* of a formula F is defined as $\#cl(F) - \#var(F)$, i.e., the difference between the number of clauses and the number of variables of F, denoted by $d(F)$. It is well-known that F is not minimal unsatisfiable if $d(F) \leqslant 0$[11,12]. So, we denote $MU(k)$ as the set of minimal unsatisfiable formulas with deficiency $k \geqslant 1$. It is well-known that *Horn-MU*, the set of minimal unsatisfiable Horn formulas, is a subset of $MU(1)$[12]. Whether or not a formula belongs to $MU(k)$ for fixed k can be decided in polynomial time[13]. It has been proved in [14] that for any $k, t \geqslant 1$ and any formula $F \in MU(t)$, there exists a formula H in $MU(k)$ and a homomorphism φ from H to F such that $\varphi(H) = F$. Moreover, for fixed $k, t \geqslant 1$ the formula H and the homomorphism φ can be constructed in polynomial time. For classes $\mathcal{C}_1$ and $\mathcal{C}_2$ of *CNF* formulas we consider the following problem.

Problem: *HOM*-$(\mathcal{C}_1, \mathcal{C}_2)$

Instance: Given formulas $H \in \mathcal{C}_1$ and $F \in \mathcal{C}_2$

Query: Does there exist a homomorphism φ from H to F such that $\varphi(H) = F$?

H. Kleine Büning and Dao-Yun Xu have proved that the problems *HOM*-(*Horn-MU*, *Horn-MU*) and *HOM*-$(MU(k), MU(t))$ are *NP*-complete[8,15].

A graph $G = (V, E)$ is isomorphic to a graph $G' = (V', E')$ if there is a bijection φ from V to V' such that for any $u, v \in V$, $(u, v) \in E$ if and only if $(\varphi(u), \varphi(v)) \in E'$. The graph isomorphism problem for undirected graphs, which is denoted by *GI*, is in *NP*. But it is an open problem whether *GI* is *NP*-complete[16]. A graph $G = (V, E)$ is homomorphic to a graph $G' = (V', E')$ if there is a mapping φ from V to V' such that for any $u, v \in V$, $(u, v) \in E$ implies $(\varphi(u), \varphi(v)) \in E'$. The homomorphism on graphs is closely relevant to the coloring of graphs. Let G and K be undirected graphs. We say that G is K-coloring if there exists a homomorphism from G to K. In [17], Hell and Nešetřil showed that if K is bipartite then the K-coloring problem is solvable in polynomial time and if K is not bipartite then the K-coloring problem is *NP*-complete. K_n denotes a complete graph with n vertices. Clearly, K_n is not bipartite for $n \geqslant 3$, whence K_n-coloring problem is *NP*-complete for $n \geqslant 3$.

A definite Horn clause $(a \vee \neg a_1 \vee \cdots \vee \neg a_m)$ can be written as $a \leftarrow a_1, \dots, a_m$ $(m \geqslant 0)$ which is called definite program clause, where $a, a_1, \dots, a_m$ are atoms of the language $\mathcal{L}$. For a definite program clause $C = a \leftarrow a_1, \dots, a_m$, we call the atom a as *head* of C, denoted by $head(C)$, and the set $\{a_1, \dots, a_m\}$ as *body* of C, denoted by $body(C)$. We denote $Atoms(C) = \{a, a_1, \dots, a_m\}$. A *definite logic program* P consists of clauses of form $a \leftarrow a_1, \dots, a_m$. We denote $Atoms(P) = \bigcup_{C \in P} Atoms(C)$.

We write $A \leqslant_p B$, if the class A is polynomially many-one reducible to the class B. $A \equiv_p B$ is an abbreviation for $A \leqslant_p B$ and $B \leqslant_p A$.

In this paper, we investigate the following decision problems.

Problem 1 (*HOM-LP*)

Instance: Given definite logic programs P and P';

Query: Does there exist a homomorphism φ from P to P' such that $\varphi(P) = P'$?

Problem 2 (*ISO-LP*)

Instance: Given definite logic programs P and P';

Query: Does there exist an isomorphism φ from P to P'?

By the *NP*-completeness of K_3-coloring, we show that the problem *HOM-LP* is *NP*-complete. By the transformation between a logic program and a graph, we show that the problem *ISO-LP* is as hard as *GI*.

2 Homomorphism for LP

Graph theoretic terminology not defined here can be found in [18]. For a graph G, we denote the set of vertices (resp. edges) by $V(G)$ (resp. $E(G)$). (u, v) (resp. $\langle u, v \rangle$) is an edge of an undirected (resp. directed) graph. A graph G is *simple* if G contains no multi-edge and self-loop. A graph G is called to be 3-*colorable* if the set $V(G)$ of vertices can be colored by at most three different colors and any two adjacent vertices are colored by different colors. If G is 3-colorable and G contains a complete subgraph K_3, then G can be colored by exactly three different colors. Clearly, a graph G is 3-colorable if and only if G is homomorphic to K_3. Please note that the problem K_3-*coloring* is *NP*-complete since K_3 is not bipartite.

Theorem 1. *The problem HOM-LP is NP-complete.*

Proof. Clearly, the problem *HOM-LP* is in *NP*. It is known that the problem K_3-coloring is NP-complete. Thus, it suffices to prove that the problem K_3-coloring can be reduced to the problem *HOM-LP*.

Let $G = (V, E)$ be a simple connected undirected graph, i.e., it is a connected undirected graph without multi-edges and self-loops, where $V = \{x_1, \dots, x_n\}$ and $E = \{(x_{i_1}, x_{j_1}), \dots, (x_{i_m}, x_{j_m})\}$, and let $K_3 = (V_0, E_0)$ be a complete graph with three vertices, where $V_0 = \{a, b, c\}$ and $E_0 = \{(a, b), (a, c), (b, c)\}$ and $\{a, b, c\} \cap \{x_1, \dots, x_n\} = \emptyset$. We define the graph $G^+ = G + K_3$. Clearly, G is 3-colorable if and only if the vertices of G^+ can be colored by exactly three different colors.

We can view vertices as variables (atoms) and transform an edge (x_i, x_j) as a clause $(\neg x_i \vee \neg x_j)$. We introduce $m+3$ new variables $y_1, y_2, y_3, z_1, \dots, z_m$ and define the following definite logic programs:

$P_0 = \{a \leftarrow; b \leftarrow; c \leftarrow; y_1 \leftarrow a, b; y_2 \leftarrow a, c;\ y_3 \leftarrow b, c\}$ and

$P = \{a \leftarrow; b \leftarrow; c \leftarrow; x_1 \leftarrow; \dots; x_n \leftarrow; y_1 \leftarrow a, b; y_2 \leftarrow a, c;\ y_3 \leftarrow b, c;\ z_1 \leftarrow x_{i_1}, x_{j_1}; \dots; z_m \leftarrow x_{i_m}, x_{j_m}\}$.

For the sake of readability, we write P as the following matrix:

$$
\begin{array}{c}
y_1 \\ y_2 \\ y_3 \\ z_1 \\ \vdots \\ z_m \\ a \\ b \\ c \\ x_1 \\ \vdots \\ x_n
\end{array}
\left(
\begin{array}{cccccccccccc}
-1 & & & & & & & & & & & \\
0 & -1 & & & & & & & & & & \\
0 & 0 & -1 & & & & & & & & & \\
0 & 0 & 0 & -1 & & & & & & & & \\
\vdots & \vdots & \vdots & & \ddots & & & & & & & \\
0 & 0 & 0 & & & -1 & & & & & & \\
1 & 1 & 0 & 0 & \cdots & 0 & -1 & & & & & \\
1 & 0 & 1 & 0 & \cdots & 0 & 0 & -1 & & & & \\
0 & 1 & 1 & 0 & \cdots & 0 & 0 & 0 & -1 & & & \\
0 & 0 & 0 & & & & 0 & & & -1 & & \\
\vdots & \vdots & \vdots & & \boldsymbol{M}(G) & & \vdots & & & & \ddots & \\
0 & 0 & 0 & & & & 0 & & & & & -1
\end{array}
\right)
$$

where the submatrix $\boldsymbol{M}(G)$ is the incidence matrix of the graph G. The k-th column of $\boldsymbol{M}(G)$ is associated with the edge (x_{i_k}, x_{j_k}) of G for $(1 \leqslant k \leqslant m)$. In the above matrix, a column of the matrix corresponds to a definite logic program clause. The sign -1 (resp. 1) means that the corresponding variable occurs in the head (resp. body) of the definite logic program clause. Clearly, P can be constructed in polynomial time on the size of G.

Now we show that G^+ can be colored by exactly three different colors if and only if there exists a homomorphism φ from P to P_0 such that $\varphi(P) = P_0$.

($\Rightarrow$) Assume that G^+ can be colored by exactly three different colors. Then the vertices a, b, and c must be colored by different colors. Suppose that the vertex a (resp. b and c) is colored by the color *a_color* (resp. *b_color* and *c_color*). By the colors of vertices, the set $V = \{x_1, \dots, x_n\}$ is parted into three disjoint subsets as follows:

$$
\begin{aligned}
V_a &= \{x \in V | \text{the vertex } x \text{ is colored by } a\_color\}, \\
V_b &= \{x \in V | \text{ the vertex } x \text{ is colored by } b\_color\} \\
V_c &= \{x \in V | \text{ the vertex } x \text{ is colored by } c\_color\}.
\end{aligned}
$$

For every $1 \leqslant k \leqslant m$, the vertices x_{i_k} and x_{j_k} must be colored by different colors since $(x_{i_k}, x_{j_k}) \in E(G)$. In other words, the vertices x_{i_k} and x_{j_k} are colored by one of the color groups $\{\{a\_color, b\_color\}, \{a\_color, c\_color\}, \{b\_color, c\_color\}\}$ only.

We define a part mapping over the set of x-variables φ_X: $\{x_1, \dots, x_n\} \rightarrow \{a, b, c\}$ as follows:

$$
\varphi_X(x) = \begin{cases} a, & \text{if } x \in V_a, \\ b, & \text{if } x \in V_b, \\ c, & \text{if } x \in V_c. \end{cases}
$$

Please note that the edge (x_{i_k}, x_{j_k}) corresponds to a definite program clause $z_k \leftarrow x_{i_k}, x_{j_k}$ $(1 \leqslant k \leqslant m)$.

By the mapping φ_X, the index set $\{1, \dots, m\}$ of the set $\{z_1 \leftarrow x_{i_1}, x_{j_1}, \dots, z_m \leftarrow x_{i_m}, x_{j_m}\}$ will be divided into three disjoint subsets as follows:

$$
\begin{aligned}
I_{ab} &= \{k | \ \varphi_X(\{x_{i_k}, x_{j_k}\}) = \{a, b\}\}, \\
I_{ac} &= \{k | \ \varphi_X(\{x_{i_k}, x_{j_k}\}) = \{a, c\}\}, \\
I_{bc} &= \{k | \ \varphi_X(\{x_{i_k}, x_{j_k}\}) = \{b, c\}\}.
\end{aligned}
$$

We define a part mapping over the set of z-variables φ_Z: $\{z_1, \dots, z_m\} \rightarrow \{y_1, y_2, y_3\}$ as follows:

$$
\varphi_Z(z_k) = \begin{cases} y_1, & \text{if } k \in I_{ab}, \\ y_2, & \text{if } k \in I_{ac}, \\ y_3, & \text{if } k \in I_{bc}. \end{cases}
$$

We now define a mapping φ from $Atoms(P)$ to $Atoms(P_0)$ by φ_X and φ_Z as follows:

$$
\varphi(x) = \begin{cases} xm, & \text{if } x \in \{y_1, y_2, y_3, a, b, c\}, \\ \varphi_X(x)m, & \text{if } x \in \{x_1, \dots, x_n\}, \\ \varphi_Z(x)m, & \text{if } x \in \{z_1, \dots, z_m\}. \end{cases}
$$

It is easy to check that φ is a homomorphism from P to P_0 and $\varphi(P) = P_0$.

($\Leftarrow$) Conversely, assume that there exists a homomorphism φ from P to P_0 such that $\varphi(P) = P_0$. By the structures of P_0 and P, for every definite program clause $z_k \leftarrow x_{i_k}, x_{j_k}$ in P, we have that $\varphi(z_k \leftarrow x_{i_k}, x_{j_k})$ is one of $\{y_1 \leftarrow a, b,\ y_2 \leftarrow a, c,\ y_3 \leftarrow b, c\}$. And we have $\varphi(x) \in \{a, b, c\}$ for every $x \in V = \{x_1, \dots, x_n\}$, and $\varphi(z) \in \{y_1, y_2, y_3\}$ for every $z \in \{z_1, \dots, z_m\}$.

We now define three subsets of V as follows:

$$
\begin{aligned}
V_a &= \{x \in V | \ \varphi(x) = a\}, \\
V_b &= \{x \in V | \ \varphi(x) = b\}, \\
V_c &= \{x \in V | \ \varphi(x) = c\}.
\end{aligned}
$$

Clearly, the three sets form a partition of V. Moreover, for each edge $(x_{i_k}, x_{j_k}) \in E(G)$ and the set $V' \in \{V_a, V_b, V_c\}$, if $x_{i_k} \in V'$ then $x_{j_k} \notin V'$, and if $x_{j_k} \in V'$ then $x_{i_k} \notin V'$. Thus, the vertices x_{i_k} and x_{j_k} are colored by different colors. Therefore, the graph G^+ can be colored by exactly three different colors. □

3 Isomorphism for LP

In this section, we consider complexity of the decision problem on isomorphism for definite logic programs. We will show that the problem is equivalent to the graph isomorphism problem (GI).

Theorem 2. $GI \leqslant_p$ *ISO-LP*.

Proof. We can associate in polynomial time with any simple undirected graph G a definite logic program P_G, such that for every pair G_1 and G_2 of graphs we have: G_1 is isomorphic to G_2 if and only if P_{G_1} is isomorphic to P_{G_2}.

Let G be a simple undirected graph with $V(G) = \{x_1, \dots, x_n\}$ and $E(G) = \{(x_{i_1}, x_{j_1}), \dots, (x_{i_m}, x_{j_m})\}$.

We can regard vertices as variables and transform an edge (x_i, x_j) as a clause $(\neg x_i \vee \neg x_j)$. By introducing m new variables $z_1, \ldots, z_m$, we define a definite logic program: $P_G = \{x_1 \leftarrow; \ldots; x_n \leftarrow; z_1 \leftarrow x_{i_1}, x_{j_1}; \ldots; z_m \leftarrow x_{i_m}, x_{j_m}\}$. P_G can be represented by the following matrix:

$$\begin{matrix} z_1 \\ \vdots \\ z_m \\ x_1 \\ \vdots \\ x_n \end{matrix} \begin{pmatrix} -1 & & & & & \\ & \ddots & & & & \\ & & -1 & & & \\ \cdot & \cdots & \cdot & -1 & & \\ \vdots & \boldsymbol{M}(G) & \vdots & & \ddots & \\ & & & & & -1 \end{pmatrix}$$

where the submatrix $\boldsymbol{M}(G)$ is the incidence matrix of the graph G. The k-th column of $\boldsymbol{M}(G)$ is associated with the edge (x_{i_k}, x_{j_k}) of G for $(1 \leqslant k \leqslant m)$. Clearly, P_G can be constructed in polynomial time on the size of G.

For given graphs G_i $(i = 1, 2)$ with n_i vertices and m_i edges, we can transform G_1 and G_2 into definite logic programs P_{G_1} and P_{G_2} respectively. By the structures of P_{G_1} and P_{G_2}, we have

$$\begin{aligned} P_{G_1} = \{&x_1^{(1)} \leftarrow; \ldots; x_{n_1}^{(1)} \leftarrow; z_1^{(1)} \leftarrow x_{i_1}^{(1)}, x_{j_1}^{(1)}; \ldots; \\ &z_{m_1}^{(1)} \leftarrow x_{i_{m_1}}^{(1)}, x_{j_{m_1}}^{(1)}\} \\ P_{G_2} = \{&x_1^{(2)} \leftarrow; \ldots; x_{n_2}^{(2)} \leftarrow; z_1^{(2)} \leftarrow x_{i_1}^{(2)}, x_{j_1}^{(2)}; \ldots; \\ &z_{m_2}^{(2)} \leftarrow x_{i_{m_2}}^{(2)}, x_{j_{m_2}}^{(2)}\}. \end{aligned}$$

Suppose ϕ is an isomorphism from G_1 to G_2. We have $n_1 = n_2 = n$, $m_1 = m_2 = m$ and a permutation π_x over $\{1, 2, \ldots, n\}$ and a permutation π_z over $\{1, 2, \ldots, m\}$, such that $\phi(x_p^{(1)}) = x_{\pi_x(p)}^{(2)}$ for each $1 \leqslant p \leqslant n$, and $(x_{i_q}^{(1)}, x_{j_q}^{(1)}) \in E(G_1)$ if and only if $(x_{\pi_x(i_{\pi_z(q)})}^{(2)}, x_{\pi_x(j_{\pi_z(q)})}^{(2)}) \in E(G_2)$ for each $1 \leqslant q \leqslant m$.

We now define an isomorphism ϕ^* from P_{G_1} to P_{G_2} as follows:

$$\phi^*(x) = \begin{cases} x_{\pi_x(p)}^{(2)}, & \text{if } x = x_p^{(1)} \in \{x_1^{(1)}, \ldots, x_n^{(1)}\}, \\ z_{\pi_z(q)}^{(2)}, & \text{if } x = z_q^{(1)} \in \{z_1^{(1)}, \ldots, z_m^{(1)}\}. \end{cases}$$

Conversely, suppose φ is an isomorphism from P_{G_1} to P_{G_2}. We have $n_1 = n_2 = n$, $m_1 = m_2 = m$ and a permutation π'_x over $\{1, 2, \ldots, n\}$ and a permutation π'_z over $\{1, 2, \ldots, m\}$ such that for each $1 \leqslant p \leqslant n$, $\varphi(x_p^{(1)}) = x_{\pi'_x(p)}^{(2)}$ and for each $1 \leqslant q \leqslant m$, $\varphi(z_q^{(1)}) = z_{\pi'_z(q)}^{(2)}$. Hence, the restriction of φ over $\{x_1, \ldots, x_n\}$ is an isomorphism from G_1 to G_2. □

Theorem 3. *ISO-LP* $\leqslant_p$ *GI*.

Proof. We can associate in polynomial time with any definite logic program P a directed graph G_P, such that for every pair P_1 and P_2 of definite logic programs we have: P_1 is isomorphic to P_2 if and only if G_{P_1} is isomorphic to G_{P_2}.

Let $P = \{C_1, \ldots, C_m\}$ be a definite logic program over the variables $x_1, \ldots, x_n$ where the clauses are given in an arbitrary but fixed order. We associate with P the directed graph $G_P = (V_P, E_P)$ as follows:

1) $V_P = V_{var} \cup V_{cl}$, where $V_{var} = \{x_1, \ldots, x_n\}$ (corresponding to variables in P), $V_{cl} = \{c_1, \ldots, c_m\}$ (corresponding to clauses in P);

2) $E_P = E_{loop} \cup E_{head} \cup E_{body}$, where $E_{loop} = \{\langle x_k, x_k \rangle \mid 1 \leqslant k \leqslant n\}$, $E_{head} = \{\langle c_i, x_k \rangle \mid x_k = head(C_i)$ for $1 \leqslant k \leqslant n$ and $1 \leqslant i \leqslant m\}$, and $E_{body} = \{\langle x_k, c_i \rangle \mid x_k \in body(C_i)$ for $1 \leqslant k \leqslant n$ and $1 \leqslant i \leqslant m\}$.

Please note that:

(a) we construct a self-loop at each vertex x_k to distinguish x_k and c_i;

(b) the edge $\langle c_i, x_k \rangle$ in E_{head} corresponds to the variable x_k occurring positively in the clauses C_i;

(c) the edge $\langle x_k, c_i \rangle$ in E_{body} corresponds to the variable x_k occurring negatively in the clauses C_i.

W.l.o.g., we assume that both P_1 and P_2 are definite logic programs with $Atoms(P_1) = Atoms(P_2) = \{x_1, \ldots, x_n\}$, $P_1 = \{C_1, \ldots, C_m\}$ and $P_2 = \{C'_1, \ldots, C'_m\}$. If otherwise, P_1 is not isomorphic to P_2. Let G_{P_1} and G_{P_2} be the associated directed graphs of P_1 and P_2, respectively, where $V(G_{P_t}) = V_{var} \cup V_{cl}^{(t)}$ $(t = 1, 2)$, $V_{cl}^{(1)} = \{c_1, \ldots, c_m\}$ and $V_{cl}^{(2)} = \{c'_1, \ldots, c'_m\}$.

Suppose that φ is an isomorphism from P_1 to P_2. We have a permutation π_v over $\{1, \ldots, n\}$ and a permutation π_c over $\{1, \ldots, m\}$ such that $\varphi(x_k) = x_{\pi_v(k)}$ for $1 \leqslant k \leqslant n$ and $\varphi(C_i) = C'_{\pi_c(i)}$ for $1 \leqslant i \leqslant m$.

We now define an isomorphism ψ from G_{P_1} to G_{P_2} as follows:

$$\psi(x) = \begin{cases} x_{\pi_a(k)}, & \text{if } x = x_k \in \{x_1, \ldots, x_n\}, \\ c'_{\pi_c(i)}, & \text{if } x = c_i \in \{c_1, \ldots, c_m\}. \end{cases}$$

Conversely, if there exists an isomorphism ψ from G_{P_1} and G_{P_2}, then we have $\psi(V_{var}) = V_{var}$. Thus, the restriction $\psi|_{V_{var}}$ is the desired isomorphism from P_1 to P_2, since $x_k = head(C_i)$ if and only if G_{P_1} contains the edge $\langle c_i, x_k \rangle$, and $x_k \in body(C_i)$ if and only if G_{P_1} contains the edge $\langle x_k, c_i \rangle$.

It is easy to prove that the isomorphism problems for undirected graph and directed graph are equivalent. Thus, we have that the problem *ISO-LP* is reducible polynomially to *GI*. □

By Theorems 2 and 3, we have *ISO-LP* $\equiv_p$ *GI*.

4 Conclusions

We investigated the transformations between a definite logic program and a graph. By the *NP*-completeness of the problem K_3-coloring, we proved that the homomorphism problem for definite logic programs is *NP*-complete, and the isomorphism problem for definite logic programs is equivalent to the graph isomorphism problem (*GI*).

References

[1] Cook S A, Reckhow R A. The relative efficiency of propositional proof system. *Journal of Symbolic Logic*, 1979, 44(1): 36–50, 2001.

[2] Krishnamurthy B. Short proofs for tricky formulas. *Acta Informatica*, 1985, 22: 253–275.

[3] Szeider S. How to Prove Unsatisfiability by Homomorphisms. Elsevier Preprint.

[4] Szeider S. NP-completeness of refutability by literal-once resolution. *Lecture Notes in Artificial Intelligence 2083*, Springer Verlag, Draft version, 2001.

[5] Urquhart A. The complexity of propositional proofs. *The Bulletin of Symbolic Logic*, 1995, 1(4): 425–467.

[6] Urquhart A. The symmetry rule in propositional logic. *Discrete Applied Mathematics*, 1999, 96–97: 177–193.

[7] Davis M, Putnam H. A computing procedure for quantification theory. *Journal of the ACM*, 1960, 7: 201–215.

[8] Daoyun Xu. On the complexity of renamings and homomorphisms for minimal unsatisfiable formulas [Dissertation]. Nanjing University, 2002.

[9] Szeider S. Homomorphisms of conjunctive normal forms. *Discrete Applied Mathematics*, 2003, 130(2): 351–356.

[10] Papadimitriou C H, Wolfe D. The complexity of facets resolved. *Journal of Computer and System Sciences*, 1988, 37(1): 2–13.

[11] Aharoni R. Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. *Journal of Combinatorial Theory, Series A*, 1996, 43(A): 196–204.

[12] G Davydov, I Davydova, H Kleine Büning. An efficient algorithm for the minimal unsatisfiability problem for a subclass of CNF. *Annals of Mathematics and Artificial Intelligence*, 1998, 23(3-4): 229–245.

[13] Fleischner H, Kullmann O, Szeider S. Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. *Theoretical Computer Science*, 2002, 289(1): 503–516.

[14] H Kleine Büning, Xishun Zhao. Polynomial time algorithms for computing a representation for minimal unsatisfiable formulas with fixed deficiency. *Information Processing Letters*, 2002, 84(3): 147–151.

[15] H Kleine Büning, Daoyun Xu. The complexity of homomorphisms and renamings for minimal unsatisfiable formulas. *Annals of Mathematics and Artificial Intelligence*, 2005, 43(1–4): 113–127.

[16] Köbler J, Schöning J, Toran J. The Graph Isomorphism Problem: Its Structural Complexity. Birkhäuser Verlag, 1993.

[17] Hell P, Nešetřil J. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 1990, 48: 92–110.

[18] Bondy J A, Murty U S R. Graph Theory with Applications. London: Macmillan, 1976.