Abstract. We propose a modal logic tailored to describe graph transformations and discuss some of its properties. We focus on a particular class of graphs called termgraphs. They are first-order terms augmented with sharing and cycles. Termgraphs allow one to describe classical data-structures (possibly with pointers) such as doubly-linked lists, circular lists etc. We show how the proposed logic can faithfully describe (i) termgraphs as well as (ii) the application of a termgraph rewrite rule (i.e. matching and replacement) and (iii) the computation of normal forms with respect to a given rewrite system. We also show how the proposed logic, which is more expressive than propositional dynamic logic, can be used to specify shapes of classical data-structures (e.g. binary trees, circular lists etc.).

1 Introduction

Graphs are common structures widely used in several areas in computer science and discrete mathematics. Their transformation constitute a domain of research per se with a large number of potential applications [11, 8, 9]. There are many different ways to define graphs and graph transformation. We consider in this paper structures known as termgraphs and their transformation via rewrite rules [5, 10]. Roughly speaking, a termgraph is a first-order term with possible sharing (of sub-terms) and cycles. Below we depict three examples of termgraphs: $G_0$ is a classical first-order term. $G_1$ represents the same expression as $G_0$ but argument $x$ is shared. $G_1$ is often used to define the function double $double(x) = G_1$. The second termgraph $G_2$ represents a circular list of two “records” (represented here by operator $\text{cons}$) sharing the same content $G_1$.

* This work has been partly funded by the project ARROWS of the French Agence Nationale de la Recherche.
Termgraphs allow to represent real-world data structures (with pointers) such as circular lists, doubly-linked lists etc [7], and rewriting allows to efficiently process such graphs. They are thus a suitable framework for declarative languages dealing with such complex data structures. However, while there exist rewriting-based proof methods for first-order terms, there is a lack of appropriate termgraph rewriting proof methods, diminishing thus their operational benefits. Indeed, equational logic provides a logical setting for first-order term rewriting [4], and many theorem provers use rewrite techniques in order to efficiently achieve equational reasoning. In [6] an extension of first-order (clausal) logic dealing with termgraphs has been proposed to give a logic counterpart of termgraph rewriting. In such a logic operations are interpreted as continuous functions [12, 13] and bisimilar graphs cannot be distinguished (two termgraphs are bisimilar if and only if they represent the same rational term). Due to that, reasoning on termgraphs is unfortunately much trickier than in first-order classical logic. For example, equational theories on termgraphs are not recursively enumerable whereas equational theories on terms are r.e.).

In this paper, we investigate a modal logic with possible worlds semantics which better fits the operational features of termgraph rewriting systems. Termgraphs can easily be interpreted within the framework of possible worlds semantics, where nodes are considered as worlds and edges as modalities. Based on this observation, we investigate a new modal logic which has been tailored to fit termgraph rewriting. We show how termgraphs as well as rewrite rules can be specified by means of modal formulae. In particular we show how a rewrite step can be defined by means of a modal formula which encodes termgraph matching (graph homomorphism) and termgraph replacement (graph construction and modification). We show also how to define properties on such structures, such as being a list, a circular list, a tree, a binary tree. The computation of termgraph normal form is formulated in this new logic. In addition, we formulate invariant preservation by rewriting rules and discuss subclasses for which validity is decidable.

The next two sections introduce respectively the considered class of termgraph rewrite systems and the proposed modal logic. In section 4 we discuss briefly the expressive power of the modal logic and show particularly how graph homomorphisms can be encoded. In section 5 we show how elementary graph transformations can be expressed as modal logic formulae whereas section 6 shows how termgraph rewriting can be specified as modal formulae. Section 7 gives some concluding remarks.
2 Termgraph Rewriting

This section defines the framework of graph rewrite systems that we consider in the paper. There are different approaches in the literature to define graph transformations. We follow here an algorithmic approach to termgraph rewriting [5]. Our definitions are consistent with [7].

Definition 21 (Graph)

A termgraph, or simply a graph is a tuple \( G = (\mathcal{N}, \mathcal{E}, \mathcal{L}^n, \mathcal{L}^e, \mathcal{S}, \mathcal{T}) \) which consists of a finite set of nodes \( \mathcal{N} \), a finite set of edges \( \mathcal{E} \), a (partial) node labelling function \( \mathcal{L}^n : \mathcal{N} \rightarrow \Omega \) which associates labels in \( \Omega \) to nodes in \( \mathcal{N} \), a (total) edge labelling function \( \mathcal{L}^e : \mathcal{E} \rightarrow \mathcal{F} \) which associates, to every edge in \( \mathcal{E} \), a label (or feature) in \( \mathcal{F} \), a source function \( \mathcal{S} : \mathcal{E} \rightarrow \mathcal{N} \) and a target function \( \mathcal{T} : \mathcal{E} \rightarrow \mathcal{N} \) which specify respectively, for every edge \( e \), its source \( \mathcal{S}(e) \) and its target \( \mathcal{T}(e) \).

Note that \( G \) is a first-order term if and only if \( G \) is a tree.

We assume that the labelling of edges \( \mathcal{L}^e \) fulfills the following additional determinism condition: \( \forall e_1, e_2 \in \mathcal{E}, (\mathcal{S}(e_1) = \mathcal{S}(e_2) \text{ and } \mathcal{L}^e(e_1) = \mathcal{L}^e(e_2)) \implies e_1 = e_2 \). This last condition expresses the fact that for every node \( n \) there exists at most one edge \( e \) of label \( a \) such that the source of \( e \) is \( n \). We denote such an edge by the tuple \((n, a, m)\) where \( m \) is the target of edge \( e \).

Notation: For each labelled node \( n \) the fact that \( \omega = \mathcal{L}^n(n) \) is written \( n : \omega \), and each unlabelled node \( n \) is written as \( n : \bullet \). This ‘unlabelled’ symbol \( \bullet \) is used in termgraphs to represent anonymous variables. \( n : \omega(a_1 \Rightarrow n_1, \ldots, a_k \Rightarrow n_k) \) describes a node \( n \) labelled by symbol \( \omega \) with \( k \) outgoing edges, \( e_1, \ldots, e_k \), such that for every edge \( e_i, \mathcal{L}^e(e_i) = a_i, \mathcal{S}(e_i) = n \) and \( \mathcal{T}(e_i) = n_i \). In the sequel we will use the linear notation of termgraphs [5] defined by the following grammar.

The variable \( A \) (resp. \( F \) and \( n \) ) ranges over the set \( \Omega \) (resp. \( \mathcal{F} \) and \( \mathcal{N} \) ):

- TermGraph ::= Node | Node + TermGraph
- Node ::= n:A(F ⇒ Node,...,F ⇒ Node)| n:• | n

the operator + stands for the disjoint union of termgraph definitions. We assume that every node is labelled at most once. The expression \( n : \omega(n_1, \ldots, n_k) \) stands for \( n : \omega(1 \Rightarrow n_1, \ldots, k \Rightarrow n_k) \).

A graph homomorphism, \( h : G \rightarrow G_1 \), where \( G = (\mathcal{N}, \mathcal{E}, \mathcal{L}^n, \mathcal{L}^e, \mathcal{S}, \mathcal{T}) \) and \( G_1 = (\mathcal{N}_1, \mathcal{E}_1, \mathcal{L}_1^n, \mathcal{L}_1^e, \mathcal{S}_1, \mathcal{T}_1) \) is a pair of functions \( h = (h^n, h^e) \) with \( h^n : \mathcal{N} \rightarrow \mathcal{N}_1 \) and \( h^e : \mathcal{E} \rightarrow \mathcal{E}_1 \) which preserves the labelling of nodes and edges as well as the source and target functions. This means that for each labelled node \( m \) in \( G \), \( \mathcal{L}_1^n(h^n(m)) = \mathcal{L}^n(m) \) and for each edge \( f \) in \( G \), \( \mathcal{L}_1^e(h^e(f)) = \mathcal{L}^e(f) \), \( \mathcal{S}_1(h^e(f)) = h^n(\mathcal{S}(f)) \) and \( \mathcal{T}_1(h^e(f)) = h^n(\mathcal{T}(f)) \). Notice that the image by \( h^n \) of an unlabelled node may be any node.

Remark: Because of the determinism condition, a homomorphism \( h : G \rightarrow G_1 \) is completely defined by the function \( h^n : \mathcal{N} \rightarrow \mathcal{N}_1 \) which should satisfy the following conditions: for each labelled node \( m \) in \( G \), \( \mathcal{L}_1^n(h^n(m)) = \mathcal{L}^n(m) \) and for every outgoing edge from \( m \), say \((m, a, w)\), for some feature \( a \) and node \( w \), the edge \((h^n(m), a, h^n(w))\) belongs to \( \mathcal{E}_1 \).
Example 22 Let $B_1$, $B_2$ and $B_3$ be the following termgraphs.

$$
\begin{array}{c}
B_1 : n_0 : h \\
\downarrow^1 \\
n_1 : g \\
\downarrow^b \\
n_2: \bullet \\
\end{array}
\quad
\begin{array}{c}
B_2 : n_0 : h \quad 1 \\
n_1 : g \quad a \\
n_2: \bullet \quad 0 \\
n_3: \bullet \\
\end{array}
\quad
\begin{array}{c}
B_3 : n_0 : h \\
\downarrow^1 \\
n_1 : g \\
\downarrow^b \\
n_2: \bullet \\
\end{array}

\quad
\begin{array}{c}
\{b\} \quad a \\
n_3: \bullet \\
\end{array}
$$

and $h$ and $h'$ be two functions on nodes defined as follows: $h(n_i) = n_i$ for $i$ in $\{0, 1, 2, 3\}$ and $h'(n_i) = n_i$ for $i$ in $\{0, 1, 2\}$ and $h'(n_3) = n_2$. $h$ defines a homomorphism from $B_1$ to $B_2$. $h'$ defines a homomorphism from $B_1$ to $B_3$. There is no homomorphism from $B_3$ to $B_2$ or to $B_1$, nor from $B_2$ to $B_1$.

The following definition introduces a notion of actions. Each action specifies an elementary transformation of graphs. These elementary actions are used later on to define graph transformations by means of rewrite rules.

Definition 23 (Actions) An action has one of the following forms.

- a node definition or node labelling $n : f(a_1 \Rightarrow n_1, \ldots, a_k \Rightarrow n_k)$ where $n, n_1, \ldots, n_k$ are nodes and $f$ is a label of node $n$. For $i \in \{1, \ldots, k\}$, $a_i$ is the label of an edge, $e_i$, such that $(L^n(e_i) = a_i)$ and whose source is $n$ ($S(e_i) = n$) and target is node $n_i$ ($T(e_i) = n_i$). This action, first creates a new node $n$ if $n$ does not already exist in the context of application of the action. Then node $n$ is defined by its label and its outgoing edges.

- an edge redirection or local redirection $n \gg_a m$ where $n, m$ are nodes and $a$ is the feature of an edge $e$ outgoing node $n$ ($S(e) = n$ and $L^e(e) = a$). This action is an edge redirection and means that the target of edge $e$ is redirected to point to the node $m$ (i.e., $T(e) = m$ after performing the action $n \gg_a m$).

- a global redirection $n \gg m$ where $n$ and $m$ are nodes. This means that all edges $e$ pointing to $n$ ($T(e) = n$) are redirected to point to the node $m$ ($T(e) = m$).

The result of applying an action $a$ to a termgraph $G = (N, E, L^n, L^e, S, T)$ is denoted by $a[G]$ and is defined as the following termgraph $G_1 = (N_1, E_1, L^n_1, L^e_1, S_1, T_1)$ such that:

- If $a = n : f(a_1 \Rightarrow n_1, \ldots, a_k \Rightarrow n_k)$ then
  - $N_1 = N \cup \{n, n_1, \ldots, n_k\}$,
  - $L^n_1(n) = f$ and $L^n_1(m) = L^n(m)$ if $m \neq n$,
  - Let $E = \{e_i \mid 1 \leq i \leq k, e_i \text{ is an edge such that } S(e_i) = n, T(e_i) = n_i \text{ and } L^e(e_i) = a_i\}$. $E_1 = E \cup E$,
  - $L^e_1(e) = \begin{cases} a_i & \text{if } e = e_i \in E \\ L^e(e) & \text{if } e \notin E \end{cases}$
\[ S_1(e) = \begin{cases} n & \text{if } e = e_i \in E \\ S(e) & \text{if } e \notin E \end{cases} \]
\[ T_1(e) = \begin{cases} n_i & \text{if } e = e_i \in E \\ T(e) & \text{if } e \notin E \end{cases} \]

∪ denotes classical union. This means that the nodes in \( \{n, n_1, \ldots, n_k\} \) which already belong to \( G \) are reused whereas the others are new.

- If \( \alpha = n \gg_\alpha m \) then
  - \( N_1 = N, \mathcal{L}^1_n = \mathcal{L}^n, \mathcal{L}^e_1 = \mathcal{L}^e, S_1 = S \) and
  - Let \( e \) be the edge of label \( a \) outgoing \( n \).
    \[ T_1(e) = \begin{cases} m & \text{if } T(e) = n \\ T(e) & \text{otherwise} \end{cases} \]

A rooted termgraph is a termgraph \( G \) with a distinguished node \( n \) called its root. We write \( G = (N, E, \mathcal{L}^n, \mathcal{L}^e, S, T, n) \). The application of an action \( \alpha \) to a rooted termgraph \( G = (N, E, \mathcal{L}^n, \mathcal{L}^e, S, T, n) \) is a rooted termgraph \( G_1 = (N_1, E_1, \mathcal{L}_1^n, \mathcal{L}_1^e, S_1, T_1, n_1) \) such that \( G_1 = \alpha[G] \) and root \( n_1 \) is defined as follows:

- \( n_1 = n \) if \( \alpha \) is not of the form \( n \gg p \).
- \( n_1 = p \) if \( \alpha \) is of the form \( n \gg p \).

The application of a sequence of actions \( \Delta \) to a (rooted) termgraph \( G \) is defined inductively as follows: \( \Delta[G] = G \) if \( \Delta \) is the empty sequence and \( \Delta[G] = \Delta'[\alpha[G]] \) if \( \Delta = \alpha; \Delta' \) where "\( ; \)" is the concatenation (or sequential) operation. Let \( h \) be a homomorphism. We denote by \( h(\Delta) \) the sequence of actions obtained from \( \Delta \) by substituting every node \( m \) occurring in \( \Delta \) by \( h(m) \).

**Example 24** This example illustrates the application of actions. Let \( H_1, H_2, H_3, H_4 \) and \( H_5 \) be the following termgraphs.

\[
\begin{align*}
H_1 : & \quad n_1 : f \\
& \downarrow^a \\
& n_2 : 0 \\
H_2 : & \quad n_1 : g \\
& \downarrow^b \\
& n_2 : 0 \quad n_3 : \bullet \\
H_3 : & \quad n_0 : h \\
& \downarrow^1 \\
& n_1 : g \\
& \downarrow^b \\
& n_2 : 0 \quad n_3 : \bullet \\
H_4 : & \quad n_0 : h \\
& \downarrow^1 \\
& n_1 : g \\
& \{e \}^a \\
& n_2 : 0 \quad n_3 : \bullet \\
H_5 : & \quad n_0 : h \\
& \downarrow^b \{1\}^a \\
& n_1 : g \\
& n_2 : 0 \quad n_3 : \bullet
\end{align*}
\]

\( H_2 \) is obtained from \( H_1 \) by applying the action \( n_1 : g \Rightarrow n_2, a \Rightarrow n_3 \). \( n_1 \) is relabelled whereas \( n_3 \) is a new unlabelled node. \( H_3 \) is obtained from \( H_2 \) by
applying the action $\alpha = n_0 : h(n_1)$. $n_0$ is a new node labelled by $h$. $h$ has one argument $n_1$. $H_4$ is obtained from $H_3$ by applying the action $n_1 \gg_a n_2$. The effect of this action is to change the target $n_3$ of the edge $(n_1, a, n_3)$ by $n_2$. $H_5$ is obtained from $H_4$ by applying the action $n_2 \gg n_0$. This action redirects the incoming edges of node $n_2$ to target node $n_0$.

**Definition 25 (Rule, system, rewrite step)** A rewrite rule is an expression of the form $l \rightarrow r$ where $l$ is a termgraph and $r$ is a sequence of actions. A rule is written $l \rightarrow (a_1, \ldots, a_n)$ or $l \rightarrow a_1; \ldots; a_n$ where the $a_i$'s are elementary actions. A termgraph rewrite system is a set of rewrite rules. We say that the term-graph $G$ rewrites to $G_1$ using the rule $l \rightarrow r$ iff there exists a homomorphism $h : l \rightarrow G$ and $G_1 = h(r)[G]$. We write $G \Rightarrow_{l \rightarrow r} G_1$, or simply $G \Rightarrow_{l} G_1$.

**Example 26** We give here an example of a rewrite step. Consider the following rewrite rule:

$$n_1 : g(a \Rightarrow n_2 : \bullet, b \Rightarrow n_3 : \bullet) \rightarrow n_0 : h(1 \Rightarrow n_1); n_1 \gg_a n_2; n_2 \gg n_0$$

The reader may easily verify that the graph $H_2$ of Example 24 can be rewritten by the considered rule into the graph $H_5$ of Example 24.

**Example 27** We give here somme illustrating examples of the considered class of rewrite systems. We first define an operation, insert, which inserts an element in a circular list.

$$r : \text{insert}(m : \bullet, p_1 : \text{cons}(m_1 : \bullet, p_1)) \rightarrow p_2 : \text{cons}(m, p_1); p_1 \gg_2 p_2; r \gg p_2$$

$$r : \text{insert}(m : \bullet, p_1 : \text{cons}(m_1 : \bullet, p_2)) + p_3 : \text{cons}(m_2, p_1) \rightarrow p_4 : \text{cons}(m, p_1); p_3 \gg_2 p_4; r \gg p_4$$

As a second example, we define below the operation length which computes the number of elements of any, possibly circular, list.

$$r : \text{length}(p : \bullet) \rightarrow r' : \text{length'}(p, p); r \gg r'$$

$$r : \text{length'}(p_1 : \text{nil}, p_2 : \bullet) \rightarrow r' : 0; r \gg r'$$

$$r : \text{length'}(p_1 : \text{cons}(n : \bullet, p_2 : \bullet), p_2) \rightarrow r' : \text{succ}(0); r \gg r'$$

$$r : \text{length'}(p_1 : \text{cons}(n : \bullet, p_2 : \bullet), p_3 : \bullet) \rightarrow r' : s(q : \bullet); q : \text{length'}(p_2, p_3); r \gg r'$$

Pointers help very often to enhance the efficiency of algorithms. In the following, we define the operation reverse which performs the so-called “in-situ list reversal”.

$$o : \text{reverse}(p : \bullet) \rightarrow o' : \text{reverse'}(p, q : \text{nil}); o \gg o'$$

$$o : \text{reverse'}(p_1 : \text{cons}(n : \bullet, q : \text{nil}), p_2 : \bullet) \rightarrow p_1 \gg_2 p_2; o \gg p_1$$

$$o : \text{reverse'}(p_1 : \text{cons}(n : \bullet, p_2 : \text{cons}(m : \bullet, p_3 : \bullet), p_4 : \bullet) \rightarrow p_1 \gg_2 p_4; o \gg_1 p_2; o \gg_2 p_1$$
The last example illustrates the encoding of classical term rewrite systems. We define the addition on naturals as well as the function double with their usual meanings.

\[
\begin{align*}
    r & : + (n : 0, m : \bullet) \rightarrow r \gg m \\
    r & : + (n : \text{succ}(p : \bullet), m : \bullet) \rightarrow q : \text{succ}(k : +(p, m)); r \gg q \\
    r & : \text{double}(n : \bullet) \rightarrow q : +(n, n); r \gg q
\end{align*}
\]

3 Modal logic

It is now time to define the syntax and the semantics of the logic of graph modifiers that will be used as a tool to talk about rooted termgraphs.

3.1 Syntax

Like the language of propositional dynamic logic, the language of the logic of graph modifiers is based on the idea of associating with each action \( \alpha \) of an action language a modal connective \( [\alpha] \). The formula \( [\alpha]\phi \) is read “after every terminating execution of \( \alpha \), \( \phi \) is true”. Consider, as in section 2, a countable set \( F \) (with typical members denoted \( a, b \), etc) of edge labels and a countable set \( \Omega \) (with typical members denoted \( \omega, \pi \), etc) of node labels. These labels are formulas defined below. A node labeled by \( \pi \) is called a \( \pi \) node.

Formally we define the set of all actions (with typical members denoted \( \alpha, \beta \), etc) and the set of all formulas (with typical members denoted \( \phi, \psi \), etc) as follows:

\[
\begin{align*}
    & - \alpha ::= a | U | n | n | \phi? | (\omega :=_g \phi) | (\omega :=_l \phi) | (a + (\phi, \psi)) | (a - (\phi, \psi)) | \\
    & \quad (\alpha; \beta) | (\alpha \cup \beta) | \alpha^*, \quad \\
    & - \phi ::= \omega | \bot | \neg \phi | (\phi \lor \psi) | [\alpha]\phi.
\end{align*}
\]

We adopt the standard abbreviations for the other Boolean connectives. Moreover, for all actions \( \alpha \) and for all formulas \( \phi \), let \( [\alpha]\phi \) be \( \neg[\alpha]\neg\phi \). As usual, we follow the standard rules for omission of the parentheses. An atomic action is either an edge label \( a \) in \( F \), the universal action \( U \), a test \( \phi? \) or an update action \( n, n, \omega :=_g \phi, \omega :=_l \phi, a + (\phi, \psi) \) or \( a - (\phi, \psi) \). \( U \) reads “go anywhere”, \( n \) reads “add some new node”, \( n \) reads “add some new node and go there”, \( \omega := g \phi \) reads “assign to \( \omega \) nodes the truth value of \( \phi \) everywhere (globally)”, \( \omega := l \phi \) reads “assign to \( \omega \) the truth value of \( \phi \) here (locally)”, \( a + (\phi, \psi) \) reads “add \( a \) edges from all \( \phi \) nodes to all \( \psi \) nodes”, and \( a - (\phi, \psi) \) reads “delete\( a \) edges from all \( \phi \) nodes to all \( \psi \) nodes”. Complex actions are built by means of the regular operators “;”, “\( \cup \)” and “\( \star^* \)”. An update action is an action without edge labels and without \( U \). An update action is \( :=_l \)-free if no local assignment \( \omega :=_l \phi \) occurs in it.
3.2 Semantics

Like the truth-conditions of the formulas of ordinary modal logics, the truth-conditions of the formulas of the logic of graph modifiers is based on the idea of interpreting, within a rooted termgraph \( G = (N, E, \mathcal{L}^n, \mathcal{L}^e, S, T, n_0) \), edge labels in \( \mathcal{F} \) by sets of edges and node labels in \( \Omega \) by sets of nodes. In this section, we consider a more general notion of node labeling functions \( \mathcal{L}^n \) of termgraphs such that nodes can have several labels (propositions). In this case the labeling function has the following profile \( \mathcal{L}^n : N \rightarrow \mathcal{P}(\Omega) \). Node labeling functions considered in section 2 where a node can have at most one label is obviously a particular case. Let \( I_G \) be the interpretation function in \( G \) of labels defined as follows:

- \( I_G(a) = \{ e \in E : \mathcal{L}^e(e) = a \} \),
- \( I_G(\omega) = \{ n \in N : \omega \in \mathcal{L}^n(n) \} \).

For all abstract actions \( a \), let \( R_G(a) = \{ (n_1, n_2) : \text{there exists an edge } e \in I_G(a) \text{ such that } S(e) = n_1 \text{ and } T(e) = n_2 \} \) be the binary relation interpreting the abstract action \( a \) in \( G \). The truth-conditions of the formulas of the logic of graph modifiers are defined by induction as follows:

- \( G \models \omega \text{ iff } n_0 \in I_G(\omega) \),
- \( G \not\models \bot \),
- \( G \models \neg \phi \text{ iff } G \not\models \phi \),
- \( G \models \phi \lor \psi \text{ iff } G \models \phi \text{ or } G \models \psi \),
- \( G \models [\alpha] \phi \text{ iff for all rooted termgraphs } G' = (N', E', \mathcal{L}'^n, \mathcal{L}'^e, S', T', n'_0), \text{ if } G \rightarrow_\alpha G' \text{ then } G' \models \phi \)

where the binary relations \( \rightarrow_\alpha \) are defined by induction as follows:

- \( G \rightarrow_\alpha G' \text{ iff } N' = N, E' = E, \mathcal{L}'^n = \mathcal{L}^n, \mathcal{L}'^e = \mathcal{L}^e, S' = S, T' = T \text{ and } (n_0, n'_0) \in R_G(\alpha) \),
- \( G \rightarrow_\phi G' \text{ iff } N' = N, E' = E, \mathcal{L}'^n = \mathcal{L}^n, \mathcal{L}'^e = \mathcal{L}^e, S' = S, T' = T, n'_0 = n_0 \text{ and } G' \models \phi \),
- \( G \rightarrow_U G' \text{ iff } N' = N, E' = E, \mathcal{L}'^n = \mathcal{L}^n, \mathcal{L}'^e = \mathcal{L}^e, S' = S, T' = T \text{ and } n'_0 = n_0 \),
- \( G \rightarrow_\alpha G' \text{ iff } N' = N \cup \{ n_1 \} \text{ where } n_1 \text{ is a new node, } E' = E, \mathcal{L}'^n(m) = \mathcal{L}^n(m) \text{ if } m \neq n_1, \mathcal{L}'^n(n_1) = \emptyset, \mathcal{L}'^e(m) = \mathcal{L}^e(m), \text{ and } L' \models \phi \text{ and } n'_0 = n_0 \text{, then } L''(m) \cup \{ \omega \} \text{ else } L''(m) \setminus \{ \omega \}, \mathcal{L}'^e = \mathcal{L}^e, S' = S, T' = T \text{ and } n'_0 = n_0 \),
- \( G \rightarrow_{\omega=\epsilon} G' \text{ iff } N' = N, E' = E, \mathcal{L}'^n(m) = \mathcal{L}^n(m) \text{ if } (N, E, \mathcal{L}^n, S, T, m) \models \phi \text{ then } L''(m) \cup \{ \omega \} \text{ else } L''(m) \setminus \{ \omega \}, \mathcal{L}'^e = \mathcal{L}^e, S' = S, T' = T \text{ and } n'_0 = n_0 \),
- \( G \rightarrow_{\alpha+(\epsilon, \psi)} G' \text{ iff } N' = N, E' = E \cup \{ (n_1, a, n_2) : (N, E, \mathcal{L}^n, \mathcal{L}^e, S, T, n_1) \models \phi \text{ and } (N, E, \mathcal{L}^n, \mathcal{L}^e, S, T, n_2) \models \psi \}, \mathcal{L}'^n = \mathcal{L}^n, \mathcal{L}'^e(e) = \text{ of } e \in E \text{ then } \mathcal{L}^e(e) = \text{ else } a, S'(e) = \text{ if } e \in E \text{ then } S(e) \text{ else } e \text{ is of the form } (n_1, a, n_2) \text{ and } S'(e) = n_1, T' = \text{ if } e \in E \text{ then } T(e) \text{ else } e \text{ is of the form } (n_1, a, n_2) \text{ and } T'(e) = n_2 \text{ and } n'_0 = n_0 \),
\[ G \rightarrow_{\alpha}(\phi, \psi) G' \text{ iff } \mathcal{N}' = \mathcal{N}, \mathcal{E}' = \mathcal{E}\setminus \{ (n_1, a, n_2) : (\mathcal{N}, \mathcal{E}, \mathcal{L}', \mathcal{T}, S, T, n_1) \models \phi \text{ and } (\mathcal{N}, \mathcal{E}, \mathcal{L}', \mathcal{S}, T, n_2) \models \psi \}, \mathcal{L}' = \mathcal{L}, \mathcal{L}'(e) = \mathcal{L}(e), S' = S, T' = T \text{ and } n_0' = n_0. \]

\[ G \rightarrow_{\alpha, \beta} G' \text{ iff there exists a rooted termgraph } G'' = (\mathcal{N}'', \mathcal{E}'', \mathcal{L}'', \mathcal{S}'', \mathcal{T}'', n_0''). \]

The above definitions of formulas reflect our intuitive understanding of the actions of the language of the logic of graph modifiers. Obviously, \( G \models (\alpha) \phi \) if there exists a rooted termgraph \( G' = (\mathcal{N}', \mathcal{E}', \mathcal{L}', \mathcal{S}', \mathcal{T}', n_0') \) such that \( G \rightarrow_{\alpha} G' \) and \( G' \models \phi \). The formula \( \phi \) is said to be valid in class \( \mathcal{C} \) of rooted termgraphs, in symbols \( \mathcal{C} \models \phi \), if \( G \models \phi \) for each rooted termgraph \( G = (\mathcal{N}, \mathcal{E}, \mathcal{L}, \mathcal{S}, \mathcal{T}, n_0) \) in \( \mathcal{C} \). The class of all rooted termgraphs will be denoted more briefly as \( \mathcal{C}_{\text{all}} \).

### 3.3 Validities

Obviously, as in propositional dynamic logic, we have

- \( \mathcal{C}_{\text{all}} \models [\phi?] \psi \leftrightarrow (\phi \rightarrow \psi) \),
- \( \mathcal{C}_{\text{all}} \models [\alpha; \beta] \phi \leftrightarrow [\alpha][\beta]\phi \),
- \( \mathcal{C}_{\text{all}} \models [\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \land [\beta]\phi \),
- \( \mathcal{C}_{\text{all}} \models [\alpha^*]\phi \leftrightarrow \phi \land [\alpha]\alpha^*\phi \).

If \( \alpha \) is a \( :=i \)-free update action then

- \( \mathcal{C}_{\text{all}} \models [\alpha] \bot \leftrightarrow \bot \),
- \( \mathcal{C}_{\text{all}} \models [\alpha] \neg \phi \leftrightarrow \neg [\alpha]\phi \),
- \( \mathcal{C}_{\text{all}} \models [\alpha](\phi \lor \psi) \leftrightarrow [\alpha]\phi \lor [\alpha]\psi \).

The next series of equivalences guarantees that each of our \( :=i \)-free update actions can be moved across the abstract actions of the form \( a \) or \( U \):

- \( \mathcal{C}_{\text{all}} \models [n][a]\phi \leftrightarrow [a][n]\phi \),
- \( \mathcal{C}_{\text{all}} \models [n][U]\phi \leftrightarrow [n]\phi \land [U][n]\phi \),
- \( \mathcal{C}_{\text{all}} \models [n][a]\phi \leftrightarrow T \),
- \( \mathcal{C}_{\text{all}} \models [n][U]\phi \leftrightarrow [n]\phi \land [U][n]\phi \),
- \( \mathcal{C}_{\text{all}} \models [\omega := g][a]\psi \leftrightarrow [a][\omega := g]\phi \),
- \( \mathcal{C}_{\text{all}} \models [\omega := g][U]\psi \leftrightarrow [U][\omega := g]\phi \),
- \( \mathcal{C}_{\text{all}} \models [a + (\phi, \psi)][b]\chi \leftrightarrow [b][a + (\phi, \psi)]\chi \text{ if } a \neq b \text{ and } \mathcal{C}_{\text{all}} \models [a + (\phi, \psi)][b]\chi \leftrightarrow [b][a + (\phi, \psi)]\chi \land (\phi \rightarrow [U]\psi \rightarrow a + (\phi, \psi)]) \text{ if } a = b \),
- \( \mathcal{C}_{\text{all}} \models [a + (\phi, \psi)][U]\chi \leftrightarrow [U][a + (\phi, \psi)]\chi \),
- \( \mathcal{C}_{\text{all}} \models [a - (\phi, \psi)][b]\chi \leftrightarrow [b][a - (\phi, \psi)]\chi \text{ if } a \neq b \text{ and } \mathcal{C}_{\text{all}} \models [a - (\phi, \psi)][b]\chi \leftrightarrow (\neg \phi \land [b][a - (\phi, \psi)]\chi \land [U][a + (\phi, \psi)]) \text{ if } a = b \),
- \( \mathcal{C}_{\text{all}} \models [a - (\phi, \psi)][U]\chi \leftrightarrow [U][a - (\phi, \psi)]\chi \).
Finally, once we have moved each of our :=-free update actions across the abstract actions of the form $a$ or $U$, these update actions can be eliminated by means of the following equivalences:

- $C_{all} \models [n] \omega \leftrightarrow \omega$,
- $C_{all} \models [n] \omega \leftrightarrow \bot$,
- $C_{all} \models [\omega :=_g \phi] \pi \leftrightarrow \pi$ if $\omega \neq \pi$ and $C_{all} \models [\omega :=_g \phi] \pi \leftrightarrow \phi$ if $\omega = \pi$,
- $C_{all} \models [a + (\phi, \psi)] \omega \leftrightarrow \omega$,
- $C_{all} \models [a - (\phi, \psi)] \omega \leftrightarrow \omega$.

**Proposition 31** For all :=-free *-free formulas $\phi$, there exists a :=-free *-free formula $\psi$ without update actions such that $C_{all} \models \phi \leftrightarrow \psi$.

**Proof.** See the above discussion.

Just as for :=-free update actions, we have the following equivalences for the update actions of the form $\omega :=_l \phi$:

- $C_{all} \models [\omega :=_l \phi] \bot \leftrightarrow \bot$,
- $C_{all} \models [\omega :=_l \phi] \neg \psi \leftrightarrow \neg [\omega :=_l \phi] \psi$,
- $C_{all} \models [\omega :=_l \phi] (\psi \lor \chi) \leftrightarrow [\omega :=_l \phi] \psi \lor [\omega :=_l \phi] \chi$,
- $C_{all} \models [\omega :=_l \phi] \pi \leftrightarrow \pi$ if $\omega \neq \pi$ and $C_{all} \models [\omega :=_g \phi] \pi \leftrightarrow \phi$ if $\omega = \pi$.

But it is not possible to formulate reduction axioms for the cases $[\omega :=_l \phi][a] \psi$ and $[\omega :=_l \phi][U] \psi$. More precisely,

**Proposition 32** There exists a *-free formula $\phi$ such that for all *-free formulas $\psi$ without update actions, $C_{all} \nvdash \phi \leftrightarrow \psi$.

**Proof.** Take the *-free formula $\phi = [\omega :=_l \bot][U][\omega :=_l \top][a] \neg \omega$. The reader may easily verify that for all rooted termgraphs $G = (N, E, L^n, L^e, S, T, n_0)$, $G \models \phi$ iff $R_G(a)$ is irreflexive. Seeing that the fact that the binary relation interpreting an abstract action of the form $a$ is irreflexive cannot be modally defined in propositional dynamic logic, then for all formulas $\psi$ without update actions, $C_{all} \nvdash \phi \leftrightarrow \psi$.

### 3.4 Decidability, axiomatization and a link with hybrid logics

Firstly, let us consider the set $L$ of all :=-free *-free formulas $\phi$ such that $C_{all} \models \phi$. Together with a procedure for deciding membership in *-free propositional dynamic logic, the equivalences preceding proposition 31 provide a procedure for deciding membership in $L$. Hence, membership in $L$ is decidable.

Secondly, let us consider the set $L(=i)$ of all *-free formulas $\phi$ such that $C_{all} \models \phi$. Aucher et al. [3] have defined a recursive translation from the language of hybrid logic [2] into the set of all our *-free formulas that preserves satisfiability. It is known that the problem of deciding satisfiability of hybrid logic formulas
is undecidable [1, Section 4.4]. The language of hybrid logic has formulas of the form $@i \phi$ (“$\phi$ is true at $i$”), $@x \phi$ (“$\phi$ is true at $x$”) and $\downarrow x. \phi$ (“$\phi$ holds after $x$ is bound to the current state”), where $NOM = \{i_1, \ldots\}$ is a set of nominals, and $SVAR = \{x_1, \ldots\}$ is a set of state variables. The (slightly adapted) translation of a given hybrid formula $\phi_0$ is recursively defined as follows.

\[
\begin{align*}
\tau(\omega) &= \omega \\
\tau(i) &= \omega_i \quad \text{where $\omega_i$ does not occur in $\phi_0$} \\
\tau(x) &= \omega_x \quad \text{where $\omega_x$ does not occur in $\phi_0$} \\
\tau(\neg \phi) &= \neg \tau(\phi) \\
\tau(\phi \lor \psi) &= \tau(\phi) \lor \tau(\psi) \\
\tau([a] \phi) &= [a] \tau(\phi) \\
\tau(\langle U \rangle \phi) &= [U] \tau(\phi) \\
\tau(\hat{x} \phi) &= \langle U \rangle (\omega_x \land \tau(\phi)) \\
\tau(\downarrow x. \phi) &= [\omega_x := g \bot][\omega_x := l \top] \tau(\phi)
\end{align*}
\]

As the satisfiability problem is undecidable in hybrid logic, membership in $L(\assign l)$ is undecidable, too.

Thirdly, let us consider the set $L^*(\assign l)$ of all $\assign l$-free formulas $\phi$ such that $C_{all} \models \phi$. It is still an open problem whether membership in $L^*(\assign l)$ is decidable or not: while the update actions can be eliminated from $\assign l$-free formulas, it is not clear whether this can be done for formulas in which e.g. iterations of assignments occur.

As for the axiomatization issue, the equivalences preceding proposition 31 provide a sound and complete axiom system of $L$, whereas no axiom system of $L(\assign l)$ and $L^*(\assign l)$ is known to be sound and complete.

4 Definability of classes of termgraphs

For all abstract actions $a$, by means of the update actions of the form $\omega \assign l \phi$, we can express the fact that the binary relation interpreting an abstract action of the form $a$ is deterministic, irreflexive or locally reflexive. More precisely, for all rooted termgraphs $G = (N, E, L^n, L^c, S, T, n_0)$,

\[- G \models [\omega := g \bot][\pi := g \bot][U][\omega := i \top][\phi : = i \top][U](\omega \rightarrow [a] \phi) \text{ iff } R_G(a) \text{ is deterministic,} \]

\[- G \models [\omega := g \bot][U][\omega := i \top][a] \text{ iff } R_G(a) \text{ is irreflexive,} \]

\[- G \models [\omega := g \bot][\omega := i \top][a] \omega \text{ iff } R_G(a) \text{ is locally reflexive in } n_0. \]

Together with the update actions of the form $\omega \assign l \phi$, the regular operation “*” enables us to define non-elementary classes of rooted termgraphs. As a first example, the class of all infinite rooted termgraphs cannot be modally defined in propositional dynamic logic but the following formula pins it down:

\[- [\omega := g \top][U; \omega ?; \omega := l \bot]^*(U) \omega. \]
As a second example, take the class of all $a$-cycle-free rooted termgraphs. It cannot be modally defined in propositional dynamic logic but the following formula pins it down:

$$- [\omega := g \top][U][\omega := i \bot][a^+]\omega.$$ 

As a third example, within the class of all $a$-deterministic rooted termgraphs, the class of all $a$-circular rooted termgraphs cannot be modally defined in propositional dynamic logic but the following formula pins it down:

$$- [\omega := g \bot][U][\omega := i \top](a^+)\omega.$$ 

Now, within the class of all rooted termgraphs that are both $a$- and $b$-deterministic, the class of all $(a \leq b)$ rooted termgraphs cannot be modally defined in propositional dynamic logic but the following formula pins it down:

$$- [\omega := g \bot][U][\omega := i \top][a]([a \cup b]^*][\pi := i \top][U](\omega \rightarrow [b]([a \cup b]^*\pi)).$$ 

Finally, within the class of all finite $(a \cup b)$-cycle-free $(a, b)$-deterministic rooted termgraphs, the class of all $(a, b)$-binary rooted termgraphs cannot be modally defined in propositional dynamic logic but the following formula pins it down:

$$- [\omega := g \bot][U][\omega := i \top][\pi := i \top]([a \cup b]^*][\pi := i \bot][U](\omega \rightarrow [b]([a \cup b]^*\pi)).$$ 

Most important of all is the ability of the language of the logic of graph modifiers to characterize finite graph homomorphisms.

**Proposition 41** Let $G = (N, E, L^n, L^e, S, T, n_0)$ be a finite rooted termgraph. There exists a *-free action $\alpha_G$ and a *-free formula $\phi_G$ such that for all finite rooted termgraphs $G' = (N', E', L'^n, L'^e, S', T', n'_0)$, $G' \models (\alpha_G)\phi_G$ iff there exists a graph homomorphism from $G$ into $G'$.

**Proof.** Let $G = (N, E, L^n, L^e, S, T, n_0)$ be a finite rooted termgraph. Suppose that $N = \{0, \ldots, N - 1\}$ and consider a sequence $(\pi_0, \ldots, \pi_{N-1})$ of pairwise distinct elements of $\Omega$. Each $\pi_i$ will identify exactly one node of $N$, and $\pi_0$ will identify the root.

We define the action $\alpha_G$ and the formula $\phi_G$ as follows:

1. $\beta_G = (\pi_0 := g \bot); \ldots; (\pi_{N-1} := g \bot)$,
2. for all non-negative integers $i$, if $i < N$ then $\gamma_i^G = (\neg \pi_0 \land \ldots \land \neg \pi_{i-1})?; (\pi_i := i \top); U$,
3. $\alpha_G = \beta_G; \gamma_0^G; \ldots; \gamma_{N-1}^G$,

3. In an $a$-circular rooted termgraph for every node $n$ there is an $i$ and there are $a_1, \ldots, a_n$ such that $a = a_1 = a_n$ and $n_k$ is related to $n_{k+1}$ by an edge labelled $a$, for all $k \leq i$.

4. Rooted termgraphs are termgraphs where the path obtained by following feature $b$ is longer than or equal to the path obtained by following feature $a$. 


be the action defined as follows:

- for all non-negative integers \( i \), if \( i < N \) then \( \psi_G^i = \text{if } L^n(i) \) is defined then \( \langle \psi \rangle^i \) else \( T \),
- for all non-negative integers \( i, j \), if \( i, j < N \) then \( \chi_G^{ij} = \text{if there exists an edge } e \in E \text{ such that } S(e) = i \) and \( T(e) = j \) then \( \langle \psi \rangle^i \) else \( T \),
- \( \phi_G = \psi_G^0 \wedge \ldots \wedge \psi_G^{N-1} \wedge \chi_G^{0,0} \wedge \ldots \wedge \chi_G^{N-1,N-1} \).

The reader may easily verify that for all finite rooted termgraphs \( G' = (N', \mathcal{E}', L'^n, L'^e, S', T') \), \( G' \models \langle \alpha_G \rangle \phi_G \) iff there exists a graph homomorphism from \( G \) to \( G' \).

5 Definability of transformations of termgraphs

In this section we show how elementary actions over termgraphs as defined in Section 2 can be encoded by means of formulas of the proposed modal logic. Let \( \alpha_n \) be the action defined as follows:

- \( \alpha_n = (\omega := g \perp); (\omega := \top); (\pi := g \perp); (\pi := g \langle a, \omega \rangle); (a - (\top, \omega)); n; (\omega := g \perp); (\omega := \top); (a + (\pi, \omega)) \).

The reader may easily verify that for all rooted termgraphs \( G = (N, \mathcal{E}, L^n, L^e, S, T, n_0) \) and \( G' = (N', \mathcal{E}', L'^n, L'^e, S', T', n'_0) \), \( G \rightarrow_{\alpha_n} G' \) iff \( G' \) is obtained from \( G \) by redirecting every \( a \)-edge pointing to the current root towards a freshly created new root. Hence, together with the update actions \( n, \omega := g \phi, \omega := \top, a + (\phi, \psi) \) and \( a - (\phi, \psi) \), the regular operations "\( \wedge \)", "\( \cup \)" and "\( \ast \)" enable us to define the elementary actions of node labelling, local redirection and global redirection of Section 2. Let us firstly consider the elementary action of node labelling: \( n : f(a_1 \Rightarrow n_1, \ldots, a_k \Rightarrow n_k) \). Applying this elementary action consists in redirecting towards nodes \( n_1, \ldots, n_k \) the targets of \( a_1, \ldots, a_k \)-edges starting from node \( n \). It corresponds to the action \( nl(n : f(a_1 \Rightarrow n_1, \ldots, a_k \Rightarrow n_k)) \) defined as follows:

- \( nl(n : f(a_1 \Rightarrow n_1, \ldots, a_k \Rightarrow n_k)) = U; \pi_n; (f := \top); (a_1 + (\pi_n, \pi_{n_1}); \ldots; (a_k + (\pi_n, \pi_{n_k})) \).

where the \( \pi_i \)'s are as in the proof of Proposition 41. The reader may easily verify that for all rooted termgraphs \( G = (N, \mathcal{E}, L^n, L^e, S, T, n_0) \), \( G' = (N', \mathcal{E}', L'^n, L'^e, S', T', n'_0) \), \( G \rightarrow_{nl(n : f(a_1 \Rightarrow n_1, \ldots, a_k \Rightarrow n_k))} G' \) iff \( G' \) is obtained from \( G \) by redirecting towards nodes \( n_1, \ldots, n_k \) the targets of \( a_1, \ldots, a_k \)-edges starting from node \( n \). Let us secondly consider the elementary action of local redirection: \( n \Rightarrow_{a}^i m \). Applying this elementary action consists in redirecting towards node \( m \) the target of an \( a \)-edge starting from node \( n \). It corresponds to the action \( lr(n, a, m) \) defined as follows:

- \( lr(n, a, m) = (a - (\pi_n, \top)); (a + (\pi_n, \pi_m)) \).

The reader may easily verify that for all rooted termgraphs \( G = (N, \mathcal{E}, L^n, L^e, S, T, n_0) \), \( G' = (N', \mathcal{E}', L'^n, L'^e, S', T', n'_0) \), \( G \rightarrow_{lr(n, a, m)} G' \) iff \( G' \) is obtained from \( G \) by redirecting towards node \( m \) the target of an \( a \)-edge starting from node \( n \). Let us
thirdly consider the elementary action of global redirection: \( n \gg \alpha \mathcal{a} m \). Applying this elementary action consists in redirecting towards node \( n \) the target of every \( a \)-edge pointing towards node \( m \). It corresponds to the action \( gr(n, a, m) \) defined as follows:

\[
- \quad gr(n, a, m) = (\lambda_a := n (a)_n); (\lambda_a := n (a)\pi_n); (a - (\pi_n)); (a + (\lambda_a, \pi_m)).
\]

The reader may easily verify that for all rooted termgraphs \( G = (N, E, L^n, L^e, S, T, n_0) \), \( G' = (N', E', L'^n, L'^e, S', T', n'_0) \), \( G \rightarrow_{gr(n, a, m)} G' \) iff \( G' \) is obtained from \( G \) by redirecting towards node \( n \) the target of every \( a \)-edge pointing towards node \( m \). To redirect towards \( n \) the target of all edges pointing towards \( m \), the action \( gr(n, a, m) \) can be performed for all \( a \in \mathcal{F} \). We get \( gr(n, m) = \bigwedge_{a \in \mathcal{F}} gr(n, a, m) \).

6 Translating rewrite rules in modal logic

Now we are ready to show how termgraph rewriting can be specified by means of formulas of the proposed modal logic.

Let \( G \rightarrow (a_1, \ldots, a_n) \) be a rewrite rule as defined in Section 2, i.e., \( G = (N, E, L^n, L^e, S, T, n_0) \) is a finite rooted termgraph and \( (a_1, \ldots, a_n) \) is a finite sequence of elementary actions. We have seen how to associate to \( G \) a \( * \)-free action \( \alpha_G \) and a \( * \)-free formula \( \phi_G \) such that for all finite rooted termgraphs \( G' = (N', E', L'^n, L'^e, S', T', n'_0) \), \( G' \models [\alpha_G; \phi_G?; \pi_1; \pi_2; \cdots; \pi_n] \) if and only if there exists a graph homomorphism from \( G \) into \( G' \). We have also seen how to associate to the elementary actions \( a_1, \ldots, a_n \) actions \( \alpha_1, \ldots, \alpha_n \). In the following proposition we show how to formulate the fact that a normal form with respect to a rewrite rule (generalization to a set of rules is obvious) satisfies a given formula \( \varphi \). A termgraph \( t \) is in normal form with respect to a rule \( R \) iff \( t \) cannot be rewritten by means of \( R \). Such formulation may help to express proof obligations of programs specified as termgraph rewrite rules. Let \( n_1, \ldots, n_k \) be the list of all nodes occurring in \( a_1, \ldots, a_n \) but not occurring in \( G \). The truth of the matter is that

**Proposition 61** Let \( \varphi \) be a modal formula. For all finite rooted termgraphs \( G' = (N', E', L'^n, L'^e, S', T', n'_0) \), every normal form of \( G' \) with respect to \( G \rightarrow (a_1, \ldots, a_n) \) satisfies \( \varphi \) iff \( G' \models [\alpha_G; \phi_G?; \pi_1; \pi_2; \cdots; \pi_n] \) for all \( a_1, \ldots, a_n \). Consider a normal form \( G'^f \) of \( G' \) with respect to \( G \rightarrow (a_1, \ldots, a_n) \). Then there exists a non-negative integer \( k \) and there exist finite rooted termgraphs \( G^0, \ldots, G^k \) such that:

\[
- \quad G^0 = G', \\
- \quad G^k = G'^f, \\
- \quad \text{for all non-negative integers } i, \text{ if } i < k \text{ then } G_i \rightarrow_{G \rightarrow (a_1, \ldots, a_n)} G_{i+1}.
\]
Hence, for all non-negative integers \( i \), if \( i < k \) then
\[
G_i \rightarrow_{\alpha_G; \phi_G'?n}(\pi_{n_1} := g \bot): (\pi_{n_1} := l \top); \ldots; (\pi_{n_k} := g \bot); (\pi_{n_k} := l \top); (\pi_{n_1} := \alpha); \ldots; (\pi_{n_k} := \alpha) G_{i+1}.
\]
Moreover, seeing that \( G^\text{nf} \) is a normal form with respect to \( G \rightarrow (a_1, \ldots, a_n) \), \( G^\text{nf} \models [\alpha_G; \phi_G'?n] \bot \). Since \( G' \models [\alpha_G; \phi_G'?n]: (\pi_{n_1} := g \bot); (\pi_{n_1} := l \top); \ldots; (\pi_{n_k} := g \bot); (\pi_{n_k} := l \top); (\pi_{n_1} := \alpha); \ldots; (\pi_{n_k} := \alpha) \), \( G^\text{nf} \models [\alpha_G; \phi_G'?n] \bot \rightarrow \varphi \), then \( G^\text{nf} \models \varphi \). Thus, every normal form of \( G' \) with respect to \( G \rightarrow (a_1, \ldots, a_n) \) satisfies \( \varphi \).

\[\Rightarrow:\) Suppose that every normal form of \( G' \) with respect to \( G \rightarrow (a_1, \ldots, a_n) \) satisfies \( \varphi \). Let \( G^\text{nf} \) be a finite rooted termgraph such that
\[
G' \rightarrow_{\alpha_G; \phi_G'?n}(\pi_{n_1} := g \bot); (\pi_{n_1} := l \top); \ldots; (\pi_{n_k} := g \bot); (\pi_{n_k} := l \top); (\pi_{n_1} := \alpha); \ldots; (\pi_{n_k} := \alpha) G^\text{nf} \models [\alpha_G; \phi_G'?n] \bot \rightarrow \varphi \).
\]

In other respects, the following proposition shows how an invariant \( \varphi \) of a rewrite rule can be expressed in the proposed logic.

**Proposition 62** Let \( \varphi \) be a modal formula. The rewrite rule \( G \rightarrow (a_1, \ldots, a_n) \) strongly preserves \( \varphi \) iff \( \models \varphi \rightarrow [\alpha_G; \phi_G'?n]: (\pi_{n_1} := g \bot); (\pi_{n_1} := l \top); \ldots; (\pi_{n_k} := g \bot); (\pi_{n_k} := l \top); (\pi_{n_1} := \alpha); \ldots; (\pi_{n_k} := \alpha) \varphi \).

**Proof.** \( \Leftarrow:\) Suppose that \( \models \varphi \rightarrow [\alpha_G; \phi_G'?n]: (\pi_{n_1} := g \bot); (\pi_{n_1} := l \top); \ldots; (\pi_{n_k} := g \bot); (\pi_{n_k} := l \top); (\pi_{n_1} := \alpha); \ldots; (\pi_{n_k} := \alpha) \varphi \). Let \( G' \), \( G'' \) be finite rooted termgraphs such that \( G' \models \varphi \) and \( G' \rightarrow_{G \rightarrow (a_1, \ldots, a_n)} G'' \). Then \( G' \models [\alpha_G; \phi_G'?n]: (\pi_{n_1} := g \bot); (\pi_{n_1} := l \top); \ldots; (\pi_{n_k} := g \bot); (\pi_{n_k} := l \top); (\pi_{n_1} := \alpha); \ldots; (\pi_{n_k} := \alpha) \varphi \) and
\[
G' \rightarrow_{\alpha_G; \phi_G'?n}(\pi_{n_1} := \alpha); (\pi_{n_1} := l \top); \ldots; (\pi_{n_k} := \alpha) G'' \models [\alpha_G; \phi_G'?n] \bot \rightarrow \varphi \) and
\[
G' \rightarrow_{\alpha_G; \phi_G'?n}(\pi_{n_1} := g \bot); (\pi_{n_1} := l \top); \ldots; (\pi_{n_k} := g \bot); (\pi_{n_k} := l \top); (\pi_{n_1} := \alpha); \ldots; (\pi_{n_k} := \alpha) G''^\text{nf} \models [\alpha_G; \phi_G'?n] \bot \rightarrow \varphi \).
\]

\( \Rightarrow:\) Suppose that the rewrite rule \( G \rightarrow (a_1, \ldots, a_n) \) strongly preserves \( \varphi \). Let \( G' \), \( G'' \) be finite rooted termgraphs such that \( G' \models \varphi \) and \( G' \rightarrow_{\alpha_G; \phi_G'?n}(\pi_{n_1} := g \bot); (\pi_{n_1} := l \top); \ldots; (\pi_{n_k} := g \bot); (\pi_{n_k} := l \top); (\pi_{n_1} := \alpha); \ldots; (\pi_{n_k} := \alpha) G'' \). Then \( G' \rightarrow_{G \rightarrow (a_1, \ldots, a_n)} G'' \rightarrow_{\alpha_G; \phi_G'?n}(\pi_{n_1} := \alpha); (\pi_{n_1} := l \top); \ldots; (\pi_{n_k} := \alpha) G'' \models [\alpha_G; \phi_G'?n] \bot \rightarrow \varphi \).

\[\varphi \rightarrow [\alpha_G; \phi_G'?n]: (\pi_{n_1} := g \bot); (\pi_{n_1} := l \top); \ldots; (\pi_{n_k} := g \bot); (\pi_{n_k} := l \top); (\pi_{n_1} := \alpha); \ldots; (\pi_{n_k} := \alpha) \varphi \).

\subsection*{7 Conclusion}

We have defined a modal logic which can be used either (i) to describe data-structures which are possibly defined by means of pointers and considered as termgraphs in this paper, (ii) to specify programs defined as rewrite rules which process these data-structures or (iii) to reason about data-structures themselves and about the behavior of the considered programs. The features of the proposed logic are very appealing. They contribute to define a logic which captures faithfully the behavior of termgraph rewrite systems. They also open new perspectives for the verification of programs manipulating pointers.

Our logic is undecidable in general. This is not surprising at all with respect to its expressive power. However, this logic is very promising in developing new
proof procedure regarding properties of termgraph rewrite systems. For instance, we have discussed a first fragment of the logic, consisting of formulas without relabelling actions, where validity is decidable. Future work include mainly the investigation of new decidable fragments of our logic and their application to program verification.

References

1. C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In J. Flum and M. Rodríguez-Artalejo, editors, Computer Science Logic, number 1683 in LNCS, pages 307–321, Madrid, Spain, 1999. Springer. Proceedings of the 8th Annual Conference of the EACSL, Madrid, September 1999.
2. C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, J. van Benthem, and F. Wolter, editors, Handbook of Modal Logic, volume 3. Elsevier Science, 2006.
3. G. Aucher, P. Balbiani, L. Fanías Del Cerro, and A. Herzig. Global and local graph modifiers. Electronic Notes in Theoretical Computer Science (ENTCS), Special issue “Proceedings of the 5th Workshop on Methods for Modalities (M4M5 2007)” , 231:293–307, 2009.
4. F. Baader and T. Nipkow. Term Rewriting and All That. Cambridge University Press, 1998.
5. H. Barendregt, M. van Eekelen, J. Glaubert, R. Kenneway, M. J. Plasmeijer, and M. Sleep. Term graph rewriting. In PARLE’87, pages 141–158. Springer Verlag LNCS 259, 1987.
6. R. Caferra, R. Echahed, and N. Peltier. A term-graph clausal logic: Completeness and incompleteness results. Journal of Applied Non-classical Logics, 18:373–411, 2008.
7. R. Echahed. Inductively sequential term-graph rewrite systems. In 4th International Conference on Graph Transformations (ICGT), volume 5214 of Lecture Notes in Computer Science, pages 84–98. Springer, 2008.
8. H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors. Handbook of Graph Grammars and Computing by Graph Transformations, Volume 2: Applications, Languages and Tools. World Scientific, 1999.
9. H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors. Handbook of Graph Grammars and Computing by Graph Transformations, Volume 3: Concurrency, Parallelism and Distribution. World Scientific, 1999.
10. D. Plump. Term graph rewriting. In H. Ehrig, G. Engels, H. J. Kreowski, and G. Rozenberg, editors, Handbook of Graph Grammars and Computing by Graph Transformation, volume 2, pages 3–61. World Scientific, 1999.
11. G. Rozenberg, editor. Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations. World Scientific, 1997.
12. J. Tiuryn. Fixed-points and algebras with infinitely long expression, part 1, regular algebras. Fundamenta Informaticae, 2:103–127, 1978.
13. J. Tiuryn. Fixed-points and algebras with infinitely long expression, part 2, μ- clones of regular algebras. Fundamenta Informaticae, 2(3):317–335, 1979.