Counterpart Semantics for a Second-Order μ-Calculus*

Fabio Gadducci[1], Alberto Lluch Lafuente[2], and Andrea Vandin[2]

[1] Department of Computer Science, University of Pisa, Italy
[2] IMT Institute for Advanced Studies Lucca, Italy

Abstract. We propose a novel approach to the semantics of quantified μ-calculi, considering models where states are algebras; the evolution relation is given by a counterpart relation (a family of partial homomorphisms), allowing for the creation, deletion, and merging of components; and formulas are interpreted over sets of state assignments (families of substitutions, associating formula variables to state components). Our proposal avoids the limitations of existing approaches, usually enforcing restrictions of the evolution relation: the resulting semantics is a streamlined and intuitively appealing one, yet it is general enough to cover most of the alternative proposals we are aware of.

Keywords: Quantified μ-calculi, counterpart semantics, graph transformation

1 Introduction

Any assessment on the usability of a visual specification formalism should rely on the existence of languages for expressing properties, as well as on the availability of tools for their verification. As far as graph transformation systems are concerned, after the seminal work of Courcelle [8], suitable variants/fragments of graph logics have been proposed and their connection with topological properties of graphs thoroughly investigated [9].

The need to reason about the possible transformations in a graph topology led more recently to the idea of combining temporal and graph logics. Before that, many authors studied decidability and complexity of temporal first-order logics, developed for reasoning about the evolution of individual components within a software system. Unfortunately, such logics are in general not decidable (see e.g. [12, 16] and the references therein). As a consequence, many efforts have been invested on defining logics (or identifying fragments) that sacrifice expressiveness in favour of efficient computability, thus providing verification tools where logics become effective specification mechanisms (see § 6).

Recent approaches [1] propose variants of quantified μ-calculi, a combination of the fix-point and modal operators of temporal logics with monadic second-order logic for graphs [8]. Albeit less expressive than full second-order proposals,

* Supported by the MIUR Project `SisteR`.

since the class of admissible predicates is restricted to first-order equality, these logics fit at the right level of abstraction for graph transformation systems: if state systems are graphs, and state components are thus graph items, one is not only interested in the topological structure of each reachable graph alone, but on its evolution as well. As a concrete example, consider graphs that represent the communication topology of a distributed leader election algorithm on a ring network that proceeds by iteratively discarding processes (edges) for the leadership. On the one hand, one would like to claim that eventually a leader will be found (the only self-closed remaining edge). This can be achieved with a formula like $\mu Z.[\exists \mathit{edge}.\mathit{source}(\mathit{edge}) = \mathit{target}(\mathit{edge}) \vee \Diamond Z]$. Note that the formula is fundamentally propositional in its temporal dimension: it asserts that at some reachable graph there will be a self-closed edge (the leader). Instead, we might be interested in expressing the eventual existence a process (an edge) which will be become the leader (self-closed). This is obtained moving to a purely first-order μ-calculus formula such as $\mu Z.\exists \mathit{edge}.[\mathit{source}(\mathit{edge}) = \mathit{target}(\mathit{edge}) \vee \Diamond Z]$.

The situation concerning the semantical models for such logics is less clearly cut. While it is obvious that a closed formula should be valued as the set of states where it holds, consider instead the open formula $\mathit{source}(\mathit{edge}) = \mathit{target}(\mathit{edge}) \vee \Diamond Z$: once the value of *edge* is chosen in the current state, how is such value passed to the states denoted by the fix-point variable Z? The issue is denoted in the quantified temporal logic literature as the *trans-world identity* problem (see [15] as well as [2] for a survey of the related philosophical issues). From a practitioner point of view, a typical solution follows the so-called "Kripke semantics" approach: roughly, a set of universal (graph) items is chosen, and its elements are used to form each state. Another solution exploits *counterpart relations*, i.e. (partial) functions among states, explicitly relating elements of different states. The first solution is the most widely adopted, and it underlines all the proposals we are aware of (as they are briefly surveyed in § 6 of the paper): it is e.g. implicitly used also in the approach discussed in [1] (admittedly the most similar in the chosen syntax/semantics to the one that we are going to introduce), since the counterpart relations used there for modelling the association of items belonging to different graphs are actually partial inclusions.

However, Kripke-like solutions do not fit too well with the possibility that items might be merged or that the evolution relation might form cycles: if the value of an open formula is a set of states, how to account for an item of a state that is first deleted and then added again? The latter problem is often solved by restricting the class of admissible evolution relations among states: this may force to reformulate the state transition relation modelling the system evolution, though such solutions tend to hamper the intuitive meaning of the logic.

In this paper we introduce a novel, purely counterpart-like semantics for quantified μ-calculi. We instantiate our proposal by considering a simple second-order syntax, and considering models where states are algebras and the evolution relation is given by a family of partial homomorphisms. Most importantly, open formulas are interpreted over sets of pairs (σ, w), for w a state and σ an assignment over w (that is, a substitution associating formula variables to components

of the state w): the resulting model thus faithfully represents also the presence of cycles in the evolution relation. Our proposal avoids the limitations of existing approaches, since it dispenses with the reformulation of the transition relation: the resulting semantics is a streamlined and intuitively appealing one, yet it is general enough to cover most of the alternatives we are aware of.

Synopsis. The opening § 2 presents our counterpart model, roughly based on [15], yet considering suitable algebras as states. Then, § 3 presents the syntax of our logic, a second-order μ-calculus that is reminiscent of the one proposed in [1]. Finally, § 4 presents the core contribution of the paper: the semantics for our logic, based on sets of assignments. The proposal is put at work with a series of simple examples in § 5. And while § 6 discusses related works, focusing on those logics applied to the verification of visual formalisms, the closing § 7 concludes the paper and outlines future research avenues.

2 Counterpart model

In this section we define the class of models over which our logic is interpreted. We begin recalling the definition of multi-sorted algebras and their homomorphisms, which lies at the basis of the structure of our worlds.

Definition 1 (Multi-sorted algebra). *A* (multi-sorted) signature Σ *is a pair* (S_Σ, F_Σ) *composed by a set of sorts* $S_\Sigma = \{\tau_1, \cdots, \tau_m\}$ *and by a set of function symbols* $F_\Sigma = \{f_\Sigma : \tau_1 \times \ldots \times \tau_n \to \tau \mid \tau_i, \tau \in S_\Sigma\}$ *typed over* S_Σ. *A* (multi-sorted) algebra $\boldsymbol{A}$ *with signature* Σ *(a* Σ*-algebra) is a pair* $(A, F^{\boldsymbol{A}}_\Sigma)$ *such that*

- *the* carrier $A = \{A_\tau \mid \tau \in S_\Sigma\}$ *is a set of elements typed over* S_Σ*;*
- $F^{\boldsymbol{A}}_\Sigma = \{f^{\boldsymbol{A}}_\Sigma : A_{\tau_1} \times \ldots \times A_{\tau_n} \to A_\tau \mid f_\Sigma : \tau_1 \times \ldots \times \tau_n \to \tau \in F_\Sigma\}$ *is a family of functions on* A *typed over* S^*_Σ.

Given two Σ*-algebras* $\boldsymbol{A}$ *and* $\boldsymbol{B}$, *a* (partial) homomorphism ϱ *is a family of possibly partial functions* $\{\varrho_\tau \mid \tau \in S_\Sigma\}$ *typed over* S_Σ, *such that for each typed function symbol* $f_\Sigma : \tau_1 \times \ldots \times \tau_n \to \tau \in F_\Sigma$ *and list of elements* $a_1, \ldots, a_n$, *if each function* ϱ_{τ_i} *is defined for the element* a_i *of type* τ_i, *then* ϱ_τ *is defined for the element* $f^{\boldsymbol{A}}_\Sigma(a_1, \ldots, a_n)$ *of type* τ *and moreover the elements* $\varrho_\tau(f^{\boldsymbol{A}}_\Sigma(a_1, \ldots, a_n))$ *and* $f^{\boldsymbol{B}}_\Sigma(\varrho_{\tau_1}(a_1), \ldots, \varrho_{\tau_n}(a_n))$ *coincide.*

Each typed function symbol $f_\Sigma \in F_\Sigma$ corresponds to a function $f^{\mathbf{A}}_\Sigma$ in $F^{\mathbf{A}}_\Sigma$: the functions in $F^{\mathbf{A}}_\Sigma$ are called fundamental operations of A. Note that our homomorphisms can be partial, possibly decreasing the domain of definition of a function and thus modelling the removal of world elements.

Example 1 (Graph algebra). As a running example we adopt a very simple unary algebra, the one for ordinary directed graphs. More precisely, the signature for directed graphs is (S_{graph}, F_{graph}). The set S_{graph} of sorts is composed by the sort of nodes τ_N and the sort of edges τ_E, while the set F_{graph} is composed by the function symbols $s : \tau_E \to \tau_N$ and $t : \tau_E \to \tau_N$ which determine respectively the

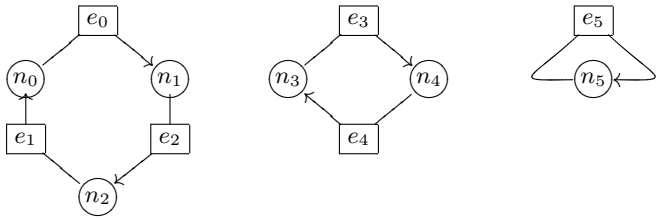

Fig. 1. Three graphs: $\mathbf{G}_0$ (left), $\mathbf{G}_1$ (middle) and $\mathbf{G}_2$ (right)

source and the target node of an edge. In Fig. 1 we find the visual representations for three graphs: $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2$. The first of these graph algebras is given by $\mathbf{G}_0 = (N_0 \uplus E_0, \{s^{\mathbf{G}_0}, t^{\mathbf{G}_0}\})$, where $N_0 = \{n_0, n_1, n_2\}$, $E_0 = \{e_0, e_1, e_2\}$, $s^{\mathbf{G}_0} = \{e_0 \mapsto n_0, e_1 \mapsto n_2, e_2 \mapsto n_1\}$ and $t^{\mathbf{G}_0} = \{e_0 \mapsto n_1, e_1 \mapsto n_0, e_2 \mapsto n_2\}$. Each graph can be understood as the state of the communication topology in a distributed algorithm: edges represent processes, source and target functions denote the ports of the processes and nodes are the communication channels.

We are interested in open terms, i.e. terms with variables. For this purpose we consider signatures Σ_X obtained by extending a multi-sorted signature Σ with a denumerable set X of variables typed over S_Σ: we let X_τ denote the τ-typed subset of variables and with x_τ or $x : \tau$ a variable with sort τ. Similarly, we let ϵ_τ or $\epsilon : \tau$ indicate a τ-sorted term.

Definition 2 (Terms). *Let Σ be a signature, let X be a (denumerable) set of individual variables typed over S_Σ, and let Σ_X denote the signature obtained extending Σ with X. The (multi-sorted) set $T(\Sigma_X)$ of* terms *obtained from Σ_X is the smallest set such that*

$$\frac{}{X \subseteq T(\Sigma_X)} \qquad \frac{\epsilon_i : \tau_i \in T(\Sigma_X),\ f : \tau_1 \times \ldots \times \tau_n \to \tau \in F_\Sigma}{f(\epsilon_1, \ldots, \epsilon_n) : \tau \ \in \ T(\Sigma_X)}$$

We omit the sort when it is clear from the context or when it is not necessary. Moreover, for an algebra $\mathbf{A}$, we let $\epsilon^{\mathbf{A}}$ denote the function associated to a term ϵ. We remark that the derived signature Σ_X does not allow to denote an individual element of the carrier directly, but only indirectly via constant symbols or variables and the well known concepts of *variable assignment* σ.

Example 2 (Terms). Consider the algebra of $\mathbf{G}_0$ and let $\{x_N, x_E\} \subseteq X$ be typed variables. Then, x_N, x_E, $s(x_E)$, and $t(x_E)$ are valid terms, while n_2 and e_1 are not. Intuitively, terms are supposed to denote either a node or an edge of the graph, but they are undefined until evaluated with respect to a chosen variable assignment. For instance, given the assignment $\sigma = \{x_E \mapsto e_1\}$ process (edge) e_1 and its *source* and *target* channels (nodes) n_2, n_0 are respectively denoted by $\sigma(x_E)$, $\sigma(s(x_E))$, and $\sigma(t(x_E))$.

We can now introduce the notion of *counterpart model*.

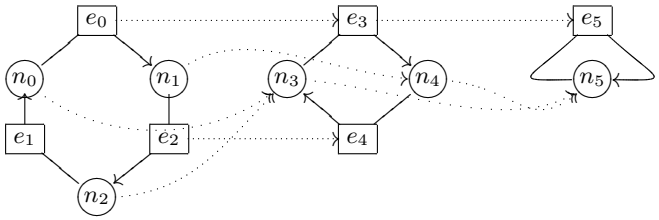

Fig. 2. A counterpart model with three sequential worlds (w_0, w_1, w_2)

Definition 3 (Counterpart model). *Let Σ be a signature, X a denumerable set of variables typed over S_Σ, and $\mathcal{A}$ the set of algebras over Σ_X. A* counterpart model *M is a tuple $(W, \rightsquigarrow, d, C)$ such that*

- *W is a non-empty set of worlds of the model;*
- *$\rightsquigarrow \subseteq (W \times W)$ is a binary relation, called* accessibility *relation over W;*
- *$d : W \to \mathcal{A}$ is a function assigning an algebra $A_{d(w)}$ to each world;*
- *$C : \rightsquigarrow \to (\mathcal{A} \rightharpoonup \mathcal{A})$ is a function assigning to every pair of worlds $(w, w') \in \rightsquigarrow$ a homomorphism (its* counterpart function *$C_{w,w'}$) from $A_{d(w)}$ to $A_{d(w')}$.*

Intuitively, the counterpart relations allow for creation, deletion, renaming and merging of elements in a type-respecting way, while forbidding duplication, i.e. it is not possible to associate an element of $A_{d(w)}$ to more than one element of $A_{d(w')}$. In the following we shall also use counterpart functions to sets of elements (with the obvious meaning of lifting the functions to sets).

Example 3 (Counterpart model). The example of Fig. 2 illustrates a model made of three worlds, namely w_0, w_1, and w_2, that are mapped into the graph algebras $\mathbf{G}_0$, $\mathbf{G}_1$, and $\mathbf{G}_2$ of Fig. 1, respectively. The transition relation is a simple sequence $w_0 \rightsquigarrow w_1 \rightsquigarrow w_2$ which can be understood as a sequential execution of the distributed leader election algorithm. The counterpart relations (drawn with dotted lines) reflect the fact that at each transition one process (edge) is discarded and its source and target channels (nodes) are merged: e_1 at the first transition and e_4 at the second one.

3 Syntax

Before presenting the syntax of our logic, we introduce the notion of second-order variables $\chi \in \mathcal{X}$. Intuitively, a variable of the second order χ_τ with sort $\tau \in S_\Sigma$ is evaluated in a set of elements of sort τ. An assignment σ associates the variables of first- (second-) order to the elements (set of elements, respectively) of the algebra $A_{d(w)}$ underlying a world w. Hence, fix-point variables $Z \in \mathcal{Z}$ range over the set of pairs (σ_w, w) relative to a counterpart model M, where w is a world of M, and σ_w an assignment for w.

Definition 4 (Quantified modal formulas). *Let Σ be a signature, $\mathcal{Z}$ a set of fix-point variables, and X, $\mathcal{X}$ (denumerable) sets of first- and second-order variables typed over S_Σ, respectively. The set $\mathcal{F}_\Sigma$ of formulas of our logic is inductively generated by the following rules*

$$\psi ::= \ tt \mid \epsilon : \tau \in_\tau \chi_\tau \mid \neg\psi \mid \psi \vee \psi \mid \exists x_\tau.\psi \mid \exists \chi_\tau.\psi \mid \Diamond\psi \mid Z \mid \mu Z.\psi$$

where $\epsilon : \tau$ is a term over Σ_X of type τ, $\in_\tau$ is a family of membership predicates typed over S_Σ indicating that (the evaluation of) a term with sort τ belongs to (the evaluation of) a second-order variable with the same sort τ, and μ denotes the least fixed point operator.

Whenever clear from the context, in the following subscripts and types will usually be omitted. We shall also use the derived symbols $\wedge$, $\rightarrow$, $\leftrightarrow$, $\forall$, as well as the modal operator $\Box$, defined as usual as $\Box\psi \equiv \neg\Diamond\neg\psi$. Moreover, as it is standard, we restrict to *monotonic* formulas, i.e., such that each fix-point variable Z occurs under the scope of an even number of negations. This is a sufficient condition for the fixed points to be well-defined.

Note that the logic is simple, yet reasonably expressive. For instance, binary equivalence can also be defined as a derived operator, namely, $\epsilon_1 : \tau =_\tau \epsilon_2 : \tau$ is defined as $\forall \chi_\tau.\ (\epsilon_1 : \tau \in_\tau \chi_\tau \leftrightarrow \epsilon_2 : \tau \in_\tau \chi_\tau)$.

Example 4 (Formula). Consider again the graph signature and our running example. The following are examples of formulas expressing different liveness properties of the distributed leader election algorithm: $\psi_1 \equiv \mu Z.(\exists x.s(x) = t(x)) \vee \Diamond Z$ (*eventually there will be a leader*) and $\psi_2 \equiv \mu Z.\exists x.(s(x) = t(x) \vee \Diamond Z)$ (*there is a process that eventually will become the leader*). Intuitively, ψ_2 is satisfied by those worlds w that can reach a world where a leader (self-closed edge) is present. Instead, ψ_1 is satisfied by those worlds w which contain a process (edge) that is a leader (i.e. its *source* and *target* ports coincide) or if this process (edge) will become a leader (self-closed) in a world reachable after some finite number of steps. Formula ψ_2 has thus quite a different meaning than ψ_1: in ψ_1 the sub-formula $\Diamond Z$ is inside the scope of the existential quantifier which fixes the element associated to x (the potential leader) in the source world to keep track of its evolution.

We now introduce the notion of context that is used for decorating terms and formulas with relevant variable-related information. For the sake of simplicity, in the rest of the paper we fix a signature Σ and denumerable sets $X, \mathcal{X}, \mathcal{Z}$ of first-order, second-order, and fix-point variables, respectively.

Definition 5 (First-order context). *A* first-order context *Γ over X is a subset of X. We write Γ, x to indicate $\Gamma \cup \{x\}$ and $\Gamma \setminus x$ to indicate $\Gamma \setminus \{x\}$.*

We indicate with $\mathcal{C}_1$ the set of all the first-order contexts. Now, we define how terms are decorated with such contexts.

Definition 6 (Term-in-context). *A* term-in-context *takes the form* $\epsilon : \tau[\Gamma]$ *where* ϵ *is a term of type* τ *over* Σ_X*, and* Γ *is a first-order context over* X*. The* set of well-formed terms-in-context $T^C(\Sigma_X)$ *over* $T(\Sigma_X)$ *is defined as*

$$\frac{x \in X_\tau, \Gamma \in \mathcal{C}_1}{x[\Gamma, x] \ \in \ T^C(\Sigma_X)} \qquad \frac{\epsilon_i : \tau_i[\Gamma] \in T^C(\Sigma_X) \ , \ f_\Sigma : \tau_1 \times \ldots \tau_n \to \tau \in F_\Sigma}{f_\Sigma(\epsilon_1, \ldots, \epsilon_n) : \tau[\Gamma] \ \in \ T^C(\Sigma_X)}$$

Example 5 (Term-in-context). Instantiating Σ with the graph signature of our running example, and considering the variables $x, y \in X_{\tau_E}$, then $s(x)[\{x\}]$ and $s(x)[\{x, y\}]$ are terms-in-context, while $s(x)[\{y\}]$ is not.

Since our logic allows for second-order quantification, we have to extend the concept of context to the second order.

Definition 7 (Second-order context). *A* second-order context Δ *over* $\mathcal{X}$ *is a subset of* $\mathcal{X}$*. The operations of addition and removal of a second-order variable with respect to a second-order context are defined as for the first-order case.*

As before, with $\mathcal{C}_2$ we indicate the set of all the second-order contexts. We can finally define how formulas are to be decorated with information about the variables involved. Their use is twofold: on the one-side, they allow for a smooth definition of the semantics, as it is going to be shown in § 4. Furthermore, even if it is not going to be further investigated here, contexts are needed to guarantee the normality of the logic and, in particular, the so-called K-scheme (see the remarks in the concluding section, as well as [2, 22] for further details).

Definition 8 (Context of a formula). *We define the* context of a formula *as a pair* $[\Gamma; \Delta]$ *where* Γ *is a first-order context, and* Δ *is a second-order context.*

Intuitively, the context of a formula has to contain at least the free variables that are actually appearing in the formula. Hence, a *formula-in-context* is a formula decorated with such a context, recursively defined as follows. In the definition we omit Σ from $\mathcal{F}_\Sigma$ considering it fixed, and use $\psi \in \mathcal{F}^{[\Gamma;\Delta]}$ as abbreviation for $\psi[\Gamma; \Delta] \in \mathcal{F}_\Sigma^{[\Gamma;\Delta]}$.

Definition 9 (Formula-in-context). *A* formula-in-context *is* $\psi[\Gamma; \Delta]$ *where* ψ *is a formula in* $\mathcal{F}_\Sigma$*, and* $[\Gamma; \Delta]$ *is a context of* ψ*. The set of well formed formulas-in-context* $\mathcal{F}_\Sigma^{[\Gamma;\Delta]}$ *over* $\mathcal{F}_\Sigma$ *is defined as*

$$\frac{}{tt \ \in \ \mathcal{F}^{[\Gamma;\Delta]}} \qquad \frac{\epsilon{:}\tau[\Gamma] \in T^{[\Gamma;\Delta]}(\Sigma_X)}{(\epsilon{:}\tau \in_\tau \chi_\tau) \ \in \ \mathcal{F}^{[\Gamma;\Delta,\chi_\tau]}} \qquad \frac{\psi \in \mathcal{F}^{[\Gamma;\Delta]}}{(\neg\psi) \ \in \ \mathcal{F}^{[\Gamma;\Delta]}}$$

$$\frac{\{\psi_1, \psi_2\} \subseteq \mathcal{F}^{[\Gamma;\Delta]}}{(\psi_1 \vee \psi_2) \ \in \ \mathcal{F}^{[\Gamma;\Delta]}} \qquad \frac{\psi \in \mathcal{F}^{[\Gamma, x_\tau;\Delta]}}{(\exists x_\tau.\psi) \ \in \ \mathcal{F}^{[\Gamma;\Delta]}} \qquad \frac{\psi \in \mathcal{F}^{[\Gamma;\Delta,\chi_\tau]}}{(\exists \chi_\tau.\psi) \ \in \ \mathcal{F}^{[\Gamma;\Delta]}}$$

$$\frac{\psi \in \mathcal{F}^{[\Gamma;\Delta]}}{(\Diamond\psi) \ \in \ \mathcal{F}^{[\Gamma;\Delta]}} \qquad \frac{}{Z \ \in \ \mathcal{F}^{[\Gamma;\Delta]}} \qquad \frac{\psi \in \mathcal{F}^{[\Gamma;\Delta]}}{(\mu Z.\psi) \ \in \ \mathcal{F}^{[\Gamma;\Delta]}}$$

where $Z \in \mathcal{Z}$*,* $x_\tau \in X$ *and* $\chi_\tau \in \mathcal{X}_\tau$*.*

Note that, by construction, a context cannot contain a variable quantified in the formula, so that e.g. a formula like $(\exists x_\tau.\psi) \vee (x_\tau =_\tau x_\tau)$ has no associated context. This property is ensured by the lack of a weakening axiom, replaced by the rule introducing any term in context $\epsilon : \tau[\Gamma]$ inside a membership expression. This solution is adopted since it simplifies the semantics for the quantifiers in Definition 12, but it is not restrictive, since the actual identity of a bound variable is immaterial. Moreover, our inference rules for formulas-in-context could anyhow be easily generalised in order to allow at least a context for any formula.

Example 6 (Formula-in-context). Consider the formula $\mu Z.\exists x.(s(x) = t(x) \vee \Diamond Z)$ proposed in Example 4. We see that it contains a quantified variable "x", but no free variables. The context of this formula can thus consist of any set of first-order variables not containing x, and of any set of second-order variables. It can even be the empty context $[\emptyset;\emptyset]$. A more useful example comes from the examination of the construction of the cited formula. Assume $[\{x\};\emptyset]$ as the context of the formula $s(x) = t(x)$. In order to apply the disjunction rule, $\Diamond Z$ has to have the same context $[\{x\};\emptyset]$. We thus obtain $(s(x) = t(x) \vee \Diamond Z)[\{x\};\emptyset]$. Now it is possible to apply the rule relative to the first-order quantifier, removing x from the context, obtaining $\exists x.(s(x) = t(x) \vee \Diamond Z)[\emptyset;\emptyset]$. Finally, applying the least fixed point rule, we obtain the formula-in-context $\mu Z.\exists x.(s(x) = t(x) \vee \Diamond Z)[\emptyset;\emptyset]$.

4 Counterpart semantics

In this section we present the core contribution of the paper: we introduce the semantical domain for our logic, and we provide the rules for evaluating a formula-in-context as a set of assignments on a counterpart model. Once more for the sake of simplicity, in the rest of the paper we fix a counterpart model M.

Definition 10 (Assignments). *A* (variable) assignment $\sigma = (\sigma_1, \sigma_2)$ *for a world $w \in M$ is a pair of partial functions typed over S_Σ such that $\sigma_1 : X \rightharpoonup A_{d(w)}$ and $\sigma_2 : \mathcal{X} \rightharpoonup 2^{A_{d(w)}}$.*

Let Ω denote the set of pairs (σ, w), for σ an assignment over the world w. A (fix-point variable) assignment *is a partial function $\rho : \mathcal{Z} \to 2^\Omega$.*

In the following, we denote by $\Omega^{[\Gamma;\Delta]}$ those pairs $((\sigma_1, \sigma_2), w)$ such that the domain of definition of σ_1 and σ_2 is Γ and Δ, respectively. Moreover, $\Omega_w \subseteq \Omega$ denotes the sub-set of assignments over a world w (i.e. those pairs whose second component is the world w), and similarly for the sub-set $\Omega_w^{[\Gamma;\Delta]} \subseteq \Omega^{[\Gamma;\Delta]}$.

Another definition regards the notions of assignment extension and restriction.

Definition 11 (Extensions and restrictions). *Let $[\Gamma;\Delta]$ be a context and $x \notin \Gamma$ a variable. Given an assignment $\sigma = (\sigma_1, \sigma_2) \in \Omega^{[\Gamma,x;\Delta]}$, its* restriction *$\sigma \downarrow_x \in \Omega^{[\Gamma;\Delta]}$ is the assignment $(\sigma_1 \downarrow_x, \sigma_2)$ obtained by restricting the domain of definition of σ_1 to Γ. Vice versa, let $a \in A_{d(w)}$ be an element of the world w.*

Given an assignment $\sigma \in \Omega_w^{[\Gamma;\Delta]}$*, its* extension $\sigma[^a/_x] \in \Omega_w^{[\Gamma,x;\Delta]}$ *is the assignment* $(\sigma_1[^a/_x], \sigma_2)$ *obtained by extending the domain of definition of* σ_1 *to* Γ, x *by assigning the element* a *to the variable* x*.*

Given a context $[\Gamma;\Delta]$ and a variable $x \notin \Gamma$, the function $2^{\downarrow_x} : 2^{\Omega^{[\Gamma,x;\Delta]}} \to 2^{\Omega^{[\Gamma;\Delta]}}$ lifts $\downarrow_x$ to sets. Vice versa, the function $\uparrow_x \colon \Omega^{[\Gamma;\Delta]} \to 2^{\Omega^{[\Gamma,x;\Delta]}}$ maps each assignment $\sigma \in \Omega_w^{[\Gamma;\Delta]}$ to the set $\{\sigma[^a/_x] \mid a \in A_{d(w)}\} \subseteq \Omega_w^{[\Gamma,x;\Delta]}$, for any world $w \in M$, and $2^{\uparrow_x} : 2^{\Omega^{[\Gamma;\Delta]}} \to 2^{\Omega^{[\Gamma,x;\Delta]}}$ denotes the lifting of $\uparrow_x$ to sets.

The corresponding functions $\downarrow_\chi$, $2^{\downarrow_\chi}$, $\uparrow_\chi$, and $2^{\uparrow_\chi}$, with respect to a second-order variable $\chi \notin \Delta$, are defined in the same way.

Example 7 (Assignments). Let us consider the counterpart model of Fig. 2, and let us denote $\lambda = (\lambda_1, \lambda_2)$ the empty assignment, regardless of the world. Then, each set $\Omega_{w_i}^{[\emptyset;\emptyset]}$ simply corresponds to $\{(\lambda, w_i)\}$, and consequently $\Omega^{[\emptyset;\emptyset]}$ corresponds to $\big\{(\lambda, w_0), (\lambda, w_1), (\lambda, w_2)\big\}$. If we extend $\Omega_{w_1}^{[\emptyset;\emptyset]}$ including the first-order variable x with sort τ_E, we obtain

$$2^{\uparrow_x}(\Omega_{w_1}^{[\emptyset;\emptyset]}) = \big\{\big((\lambda_1[^{e_3}/_x], \lambda_2), w_1\big), \big((\lambda_1[^{e_4}/_x]\lambda_2), w_1\big)\big\}$$

which is in turn equivalent to $\big\{\big((\{x \mapsto e_3\}, \lambda_2), w_1\big), \big\{\big((\{x \mapsto e_4\}, \lambda_2), w_1\big)\big\}$.

As a final step, we have to define when a given pair of assignments is compatible with the counterpart relations.

Definition 12 (Assignment counterpart). *Let* $\sigma_w \in \Omega_w$*,* $\sigma_{w'} \in \Omega_{w'}$ *be two assignments. We say that* $\sigma_{w'}$ *is a counterpart of* σ_w *relative to* $[\Gamma;\Delta]$*, denoted as* $\sigma_w \overset{[\Gamma;\Delta]}{\rightsquigarrow} \sigma_{w'}$*, if*

1. *for each first-order variable* $x \in \Gamma$*, the elements assigned to* x *by* σ_w *and by* $\sigma_{w'}$ *are in counterpart relation, i.e.* $C_{w,w'}(\sigma_w(x)) = \sigma_{w'}(x)$
2. *for each second-order variable* χ *in* Δ*, the sets assigned to* χ *by* σ_w *and by* $\sigma_{w'}$ *are in counterpart relation, i.e.* $2^{C_{w,w'}}(\sigma_w(\chi)) = \sigma'_w(\chi)$

Lifting $C_{w,w'}$ to the second-order case means that for each variable χ the set denoted by its interpretation $\sigma_w(\chi)$ is precisely mapped by the counterpart relation to the set $\sigma'_w(\chi)$.

It should be noticed that we do not accept as a valid counterpart relation one where an already assigned element is canceled. Indeed, restricting the domain of discourse to existing entities is the main characterising feature of the counterpart solutions: this aspect of our proposal is going to be illustrated by examples in § 5, while it is discussed in some detail in [2, 22].

We are now ready to introduce the semantical evaluation for our logic in a model M. It associates to a formula-in-context $\psi[\Gamma;\Delta]$ a set of assignments over the worlds of M contained in $\Omega^{[\Gamma;\Delta]}$. Hence, the domain os these assignments is exactly $\langle\Gamma;\Delta\rangle$, and thus our proposal is reminiscent of the semantics of temporal formulas over sets of constraints introduced in [13].

Definition 13 (Semantics). *Let $\psi[\Gamma;\Delta]$ be a formula-in-context. The evaluation of $\psi[\Gamma;\Delta]$ in M under the assignment $\rho : \mathcal{Z} \to 2^{\Omega^{[\Gamma;\Delta]}}$ is given by the function $[\![\cdot]\!]_\rho : \mathcal{F}^{[\Gamma;\Delta]} \to \Omega^{[\Gamma;\Delta]}$ defined as*

$$
\begin{aligned}
[\![tt[\Gamma;\Delta]]\!]_\rho &= \Omega^{[\Gamma;\Delta]} \\
[\![(\epsilon : \tau \in_\tau \chi_\tau)[\Gamma;\Delta]]\!]_\rho &= \{(\sigma, w) \in \Omega^{[\Gamma;\Delta]} \mid \sigma(\epsilon) \in \sigma(\chi_\tau)\} \\
[\![\neg\psi[\Gamma;\Delta]]\!]_\rho &= \Omega^{[\Gamma;\Delta]} \setminus [\![\psi[\Gamma;\Delta]]\!]_\rho \\
[\![\psi_1 \vee \psi_2[\Gamma;\Delta]]\!]_\rho &= [\![\psi_1[\Gamma;\Delta]]\!]_\rho \cup [\![\psi_2[\Gamma;\Delta]]\!]_\rho \\
[\![\exists x_\tau.\ \psi[\Gamma;\Delta]]\!]_\rho &= 2^{\downarrow_{x_\tau}}([\![\psi[\Gamma, x_\tau;\Delta]]\!]_{(2^{\uparrow_x} \circ \rho)}) \\
[\![\exists \chi_\tau.\ \psi[\Gamma;\Delta]]\!]_\rho &= 2^{\downarrow_{\chi_\tau}}([\![\psi[\Gamma;\Delta,\chi_\tau]]\!]_{(2^{\uparrow_\chi} \circ \rho)}) \\
[\![\Diamond\psi[\Gamma;\Delta]]\!]_\rho &= \{(\sigma, w) \in \Omega^{[\Gamma;\Delta]} \mid \exists(\sigma', w') \in [\![\psi[\Gamma;\Delta]]\!]_\rho \ .\ \sigma \overset{[\Gamma;\Delta]}{\rightsquigarrow} \sigma'\} \\
[\![Z[\Gamma;\Delta]]\!]_\rho &= \rho(Z) \\
[\![\mu Z.\psi[\Gamma;\Delta]]\!]_\rho &= \mathit{lfp}(\lambda Y.[\![\psi[\Gamma;\Delta]]\!]_\rho[{}^{Y}/_{Z}])
\end{aligned}
$$

Note that in the evaluation of the membership predicate, $\sigma(\epsilon)$ denotes the lifting of the substitution σ_1 to the set of terms over Σ_X.

In the evaluation of the quantifiers, it is pivotal to require that the assignment ρ for fix-point variables is modified, in order to account for the extensions to the newly introduced variables: it allows for a proper sorting of $\rho(Z)$, since it must now belong to the subsets of $\Omega^{[\Gamma,x;\Delta]}$ ($\Omega^{[\Gamma;\Delta,\chi]}$ in the second-order case).

Finally, in the evaluation of the modal operator the "renaming" of values across worlds is ensured by requiring that the assignments σ and σ' are in counterpart relation. Thus, our semantics discards those worlds that are reachable but are not in counterpart with respect to the current context. The rationale behind this is that it makes no sense to make claims about about non-existence (see [22, 2]). We shall illustrate this issue with some examples in § 5.

The semantics of our logic is well-defined. In particular, the restriction to formulas where all occurrences of fix-point variables are positive guarantees that any function $\lambda Y.[\![\psi[\Gamma;\Delta]]\!]_\rho[{}^{Y}/_{Z}]$ is monotonic. Therefore, by Knaster-Tarski theorem, fixed points are well-defined.

The evaluation of a closed formula, i.e. of a formula $\psi[\emptyset;\emptyset]$ with an empty context, is just a set of pairs $\{(\lambda, w)\}$, for λ the empty assignment over the world w. Hence, such an evaluation characterises a set of worlds: this ensures that our proposal properly extends the standard semantics for propositional modal logics.

Example 8 (Evaluation of a formula-in-context). We have now the means for showing the evaluation of the formula-in-context $\mu Z.\exists x.(s(x) = t(x) \vee \Diamond Z)[\emptyset;\emptyset]$ proposed in Example 6 in the model presented in Example 3. Recall that the formula states that some process (edge) will eventually become the unique leader (self-closed). According to the rule semantics we have to obtain the least fixed point of $[\![\exists x.(s(x) = t(x) \vee \Diamond Z)[\emptyset;\emptyset]]\!]_\rho$. Intuitively we expect that the formula holds for all worlds with the empty assignment, i.e. $\{(\lambda, w_0), (\lambda, w_1), (\lambda, w_2)\}$, because w_2 contains the process (edge) e_5 which is a leader (self-closed), w_1 contains process (edge) e_3 which has counterpart e_5, and w_0 has the process (edge) e_0 which has counterpart e_3. Consider the assignment $\rho = (Z, \{(\lambda, w_0), (\lambda, w_1), (\lambda, w_2)\})$.

If we apply the semantics of the existential quantifier $\rho(Z)$ becomes $\{(\{x \mapsto e_0\}, w_0), (\{x \mapsto e_1\}, w_0), (\{x \mapsto e_2\}, w_0)\} \cup \{(\{x \mapsto e_3\}, w_1), (\{x \mapsto e_4\}, w_1)\} \cup$

$\{(\{x \mapsto e_5\}, w_2)\}$. The assignment $(\{x \mapsto e_5\}, w_2)$ clearly verifies the formula $\psi_x \equiv (s(x) = t(x))$, so (λ, w_2) verifies the entire formula-in-context. Considering (λ, w_1), neither $\{x \mapsto e_3\}$ nor $\{x \mapsto e_4\}$ verify ψ_x, but $\{x \mapsto e_3\} \overset{[x,\emptyset]}{\rightsquigarrow} \{x \mapsto e_5\}$ holds, and $(\{x \mapsto e_5\}, w_2) \in \rho(Z)$. So $\{\lambda, w_1\}$ verifies the entire formula. Lastly, considering $\{\lambda, w_0\}$, as for w_1 ψ_x is not verified in w_0, but $\{x \mapsto e_0\} \overset{[x,\emptyset]}{\rightsquigarrow} \{x \mapsto e_3\}$ holds, and $(\{x \mapsto e_3\}, w_1) \in \rho(Z)$. So also $\{\lambda, w_0\}$ verifies the entire formula. Thus $\{(\lambda, w_0), (\lambda, w_1), (\lambda, w_2)\}$ is a fix-point and indeed the smallest one. Therefore, the semantics of our formula is the set of all worlds, as we intuitively expected.

5 Examples

The aim of this section is to illustrate the use of the logic to express properties of the evolution of systems and their components. Most of the examples are drawn from the literature reviewed in § 6. In order to make the syntax lighter, we do not indicate the contexts of the formulas and the types of the variables.

Death. The creation and destruction of entities has attracted the interest of various authors (e.g. [11, 23]) as a mean for reasoning about the allocation and deallocation of resources or processes. It is important to understand that our logic has no built-in mechanism for that purpose, hence it allows for various interpretations of what it means for an entity to be deleted. In our setting, an entity that has no counterpart in a certain world w' is an entity that simply does not exist in w', which is different from an entity that is deallocated (whose existence we might want to remember).

To illustrate the difference between absence and deallocation, we consider absence first and then some particular flavours of deallocation. Recall that our logic mainly regards the evolution of existing entities, thus disregarding of their absence from the system. Therefore, while predicates regarding the presence and absence of entities can be defined (e.g. $\mathbf{present}(x) \equiv \exists y.x = y$, $\mathbf{absent}(x) \equiv \neg\mathbf{present}(x)$) their semantics in our logic might not be meaningful. For instance, under the scope of the next-time modality, predicates over x should be intended as "as long as x is present", so that formulas like $\Box\mathbf{present}(x)$ might accept assignments for x in worlds that can evolve by deleting x. The key point is that our logic should be used to reason about living entities.

Now, when one is interested in reasoning about deallocated entities, one possible solution is to introduce a particular value $\perp$ in all domains and map deallocated entities (and the functions over them) onto that value (morphisms become total). This is essentially the underlying idea of [11]. Then, a deallocated entity can be characterised with predicate $\mathbf{dead}(x) \equiv x = \perp$. Several issues arise when reasoning about deallocated entities. For instance, the choice of [11] is for all predicates over deallocated entities to be false, even the trivial self equality $x = x$, which is a tautology for living entities.

Our logic is flexible enough to adopt that strategy, by e.g. redefining the abbreviation $\epsilon_1 =_\tau \epsilon_2$ for binary equality as follows $\forall\chi_\tau.\ \big((\epsilon_1 \in_\tau \chi_\tau \leftrightarrow \epsilon_2 \in_\tau$

$\chi_\tau) \wedge (\bot \notin \chi_\tau)\big)$. Other choices are also possible, like the one in [1] where the identity of deallocated entities is kept but quantification restricts to living entities.

Birth. When reasoning about entity creation, it is interesting to distinguish new from old entities. Our logic has no built-in mechanism (like e.g. in [11]) for this purpose, yet one can assume that this information is provided by the model (by using *new* and *old* values and a function from entities into those values).

Still, it is possible in general to define a modal predicate to capture the creation of a new entity x as follows: $\langle \mathbf{new}(x) \rangle(\psi) \equiv \exists \chi. \forall y. y \in \chi \wedge \Diamond \exists x. x \notin \chi \wedge \psi$. Note that the defined modality is existential and restricts to non-deleting steps only, but other choices are of course possible.

Growth. Our logic is suited for expressing properties about the growth of a system. For instance, a growth bound of 3 is stated with $\mathbf{at\text{-}most\text{-}2} \equiv \forall x. \forall y. \forall z. x = y \vee y = z \vee z = x$ as in [11] and is required as an invariant with $\nu Z. \mathbf{at\text{-}most\text{-}2} \wedge \Box Z$, for $\nu Z. -$ the (informally defined) operator for greatest fixed point.

More interestingly, we can express properties along entity preserving behaviours. For instance, an **all-preserved** modality focussing on system evolutions where no entity is deleted is $\langle \mathbf{all\text{-}preserved} \rangle(\psi) \equiv \exists \chi. \forall y. y \in \chi \wedge \Diamond \psi$.

Similarly, an abbreviation for system steps creating at least one element but preserving the rest is $\langle \mathbf{one\text{-}more} \rangle(\psi) \equiv \exists \chi. \forall y. y \in \chi \wedge \Diamond \exists x. x \notin \chi \wedge \psi$.

Finally, we can then state the *possibility* of unbounded growth with the following formula: $\nu Z. \mu Z'. \langle \mathbf{one\text{-}more} \rangle(Z') \wedge \langle \mathbf{all\text{-}preserved} \rangle(Z)$. Instead, the *necessity* of unbounded growth (see e.g. [11]) would require a model with explicit deallocation as suggested above in order to be able to require absence of deletions.

Life. Apart from the growth in the number of entities, our logic regards the evolution of those entities. A typical example is the mobility of objects (like the message propagation in the example of [1]). Assuming an algebra of objects and locations with a function *loc* for denoting the location of an object, we can express location change for an object x with predicate $\mathbf{moves}(x, \psi) \equiv \exists y. y = loc(x) \wedge \Box(loc(x) \neq y \wedge \psi)$. Then we can express that x never remains in the same location with $\nu Z. \mathbf{moves}(x, Z)$.

Along the same lines, we can define other typical individual safety and liveness properties. For instance, individual mutual exclusion (used e.g. in [23]) can be stated with formula like $\nu Z. loc(x) \neq loc(y) \wedge \Box Z$ which requires x and y never to be in the same location. Another example are individual responsiveness properties, like requiring two entities to eventually meet whenever they are in separate locations: $\nu Z. loc(x) \neq loc(y) \rightarrow (\mu Z'. loc(x) = loc(y) \vee \Diamond Z') \wedge Z$.

6 Related works

As we mentioned in the introduction, many authors addressed decidability and complexity issues concerning quantified modal logics, and many efforts have been focused on defining logics (or identifying fragments) that sacrifice expressivity in favour of efficient computability. The aim of our work is not concerned with such aspects yet, since we are interested in defining fist a generic, natural and intuitive semantics that overcomes some of the drawbacks of traditional Kripke-like semantics and follows the spirit of counterpart semantics. Thus, this section reviews some current proposals for quantified modal logics, trying to sum up the differences with the present paper, with a specific interest towards those approaches developed for the verification of visual specification formalisms.

Logics for reasoning about knowledge change (e.g. temporal description logics) have been proposed by various authors (see e.g. [12, 16]), either as first-order extensions of classical linear- and branching-time temporal logics such as LTL and CTL [16], or as extensions of the modal μ-calculus [12]. The semantics is typically given in Kripke-style with a unique domain of interpretation that does not allow for merging or renaming of elements. Decidability results are given for some fragments, e.g. the *monodic* ones, roughly consisting of equality-free formulas with a restricted number of free variables under temporal operators.

Another setting where quantified temporal logics have raised interest are graph transformation systems, where software systems exhibiting features such as component or resource allocation, de-allocation, re-allocation or fusion are conveniently modelled using graph morphisms. For instance, the approach of [1] aims at building a verification setting where graph transition systems are abstracted into a sort of Petri nets. The specification mechanism is a logic that mixes the modal μ-calculus with Courcelle's Monadic Second-Order Logic for graphs [8]. The graph transition systems considered are not allowed to introduce merging or renaming of graph items, and the semantics is defined over the *unravelling* of the graph transition system, i.e. a tree that represents the unfolded state space and that guarantees some additional properties such as no-reuse of item names. Another example can be found [14], where a graph logic was developed for encoding a spatial logic for the π-calculus [3] in a graph-based setting. The logic extends the μ-calculus with a node-binding modal operator, quantifiers and other ingredients along the ones in [8] to describe the graphical structure of configurations. Merging and renaming is allowed for some restricted cases only.

Another graph-based approach is described in [19], focusing on finite-state graph transition systems, and using a linear-time logic whose structural aspects are expressed with regular expressions over paths. The same author investigated first-order temporal logics for various structures. In [20] he proposes an extension of CTL with first- and (monadic) second-order quantification. The semantics is interpreted over *algebra automata*, i.e. automata enriched with an algebraic structure of states, and with a morphism-like transition relation that allows for renaming elements. The model checking problem over finite automata is shown reducible to the ordinary model checking of CTL formulas over Kripke struc-

tures, while preserving the necessary structure to exploit name symmetries. A similar approach is followed in [11] but based on LTL and including predicates to reason about allocation, deallocation and reallocation of objects. The notion of name-equipped automata allows for injective renaming, but forbids merging. The semantics of the next time modality does not discard accessible worlds where elements assigned to variables are deleted, but in this case the assignment becomes undefined so that the logic allows for expressing deallocation but equality predicates over undefined variables become false (even the simple case $x = x$). Instead, [10] is concerned with the approximation of special kinds of graphs and the verification of a similar logic for verifying pointer structures on a heap. Another logic to reason about the dynamics featured in object-oriented programming languages is *evolution logic* [23], a first-order version of LTL. The model checking approach focuses on abstract interpretation rather than symmetries.

Spatio-temporal logics form another track of formalisms for describing the evolution of process and data structures. Early works aimed at reasoning about networks of processes (e.g. the *multiprocess network logic* of [18]), and were based on extensions of classical linear- and branching-time logics with first-order quantifiers. In these works, the set of processes was considered to be fixed (i.e. no dynamic creation or deletion was considered) so that the elimination of quantifiers was possible. In the last years, spatial logics evolved and were mostly defined for algebraically presented systems. We cite among others spatial logics for process calculi like the π-calculus [3, 4] and mobile ambients [7], or for data structures such as graphs [5], heaps [21], and trees [6]. The common idea in such approaches is to mix temporal modalities with spatial operators that represent the dual of the operators of the algebra, like parallel (de)composition of processes or graphs, and various forms of (name) quantification. Renaming and merging of elements is typically restricted to some special cases like α-renaming and name extrusion.

7 Conclusions and further works

The present paper introduces a novel semantics for a second-order μ-calculus. With respect to other approaches, including those sketched in § 6, our proposal allows for a simple definition of the semantical universe by means of counterpart models. The idea of associating to (open) formulas sets of assignments, instead of just worlds, allows for a straightforward interpretation of fixed points and for their smooth integration with the evaluation of quantifiers, which often asked for a restriction of the class of admissible models.

The starting point for our proposal was the survey on quantified modal logic proposed by Belardinelli [2], further instantiated to graph transformations in the master's thesis of the third author [22]. The present article is a revised and extended version of the latter, taking into account also fix-point operators.

We foresee a few obvious directions for further research. As a start, we would like to investigate if the correspondence results between quantified μ-calculi and Petri nets logics proposed in [1] could be lifted to our framework, and its richer

family of counterpart relations. We would also like to better understand the relationship with spatial logics, along the lines of [14], possibly adopting a family of labelled counterpart relations, and the richer modal operators $\Diamond_{\langle p,Y\rangle}$, basically stating that the transition between worlds is caused by a specific rule p, that may create a chosen set Y of new elements. Another interesting point is in understanding the tradeoff between expressivity and complexity regarding the choice of information being discarded in the semantics of the modal operator. As we already discussed, we ignore those reachable worlds that are not in counterpart relation with respect to the current assignment, while other choices are possible like accepting the worlds, but making assignments undefined when the assigned element is deleted [11] or not discarding anything [1].

Also the development of adequate proof systems should be pursued. We did not further investigate the topic here, yet the use of formulas-in-context guarantees the so-called K-scheme, stating that if the formula $\Box(\psi_1 \to \psi_2)[\Gamma;\Delta]$ holds, then the $\Box$ operator may distribute, i.e.. also the formula $\Box(\psi_1) \to \Box(\psi_2)[\Gamma;\Delta]$ holds. The use of contexts is pivotal here, since otherwise the axiom might not always be satisfied. Instead, its validity tells us that the resulting logic is normal, which is a property of all classical modal logics [17].

Acknowledgments We are indebted to Giacomo Lenzi for his helpful comments and advices at the very early stages of our research.

References

1. Baldan, P., Corradini, A., König, B., Lluch Lafuente, A.: A temporal graph logic for verification of graph transformation systems. In: Fiadeiro, J.L., Schobbens, P.Y. (eds.) 18th International Workshop on Recent Trends in Algebraic Development Techniques (WADT'06). LNCS, vol. 4409, pp. 1–20. Springer (2007)
2. Belardinelli, F.: Quantified Modal Logic and the Ontology of Physical Objects. Ph.D. thesis, Scuola Normale Superiore of Pisa (2006)
3. Caires, L.: Behavioral and spatial observations in a logic for the π-calculus. In: Walukiewicz, I. (ed.) 7th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'04). LNCS, vol. 2987, pp. 72–87. Springer (2004)
4. Caires, L., Cardelli, L.: A spatial logic for concurrency (part I). Information and Computation 186(2), 194–235 (2003)
5. Cardelli, L., Gardner, P., Ghelli, G.: A spatial logic for querying graphs. In: Widmayer, P., Ruiz, F.T., Bueno, R.M., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) 30th International Colloquium on Automata, Languages and Programming (ICALP'02). LNCS, vol. 2380, pp. 597–610. Springer (2002)
6. Cardelli, L., Gardner, P., Ghelli, G.: Manipulating trees with hidden labels. In: Gordon, A. (ed.) 6th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'03). LNCS, vol. 2620, pp. 216–232. Springer (2003)
7. Cardelli, L., Ghelli, G.: TQL: a query language for semistructured data based on the ambient logic. Mathematical Structures in Computer Science 14(3), 285–327 (2004)

8. Courcelle, B.: The expression of graph properties and graph transformations in monadic second-order logic. In: Rozenberg, G. (ed.) Handbook of Graph Grammars and Computing by Graph Transformation, pp. 313–400. World Scientific (1997)
9. Dawar, A., Gardner, P., Ghelli, G.: Expressiveness and complexity of graph logic. Information and Compututation 205(3), 263–310 (2007)
10. Distefano, D., Katoen, J.P., Rensink, A.: Who is pointing when to whom? In: Lodaya, K., Mahajan, M. (eds.) 32nd International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'04). LNCS, vol. 3328, pp. 250–262. Springer (2004)
11. Distefano, D., Rensink, A., Katoen, J.P.: Model checking birth and death. In: Baeza-Yates, R.A., Montanari, U., Santoro, N. (eds.) 2nd IFIP International Conference on Theoretical Computer Science (TCS'02). IFIP Conference Proceedings, vol. 223, pp. 435–447. Kluwer (2002)
12. Franconi, E., Toman, D.: Fixpoint extensions of temporal description logics. In: Calvanese, D., Giacomo, G.D., Franconi, E. (eds.) 16th International Workshop on Description Logics (DL'03). CEUR Workshop Proceedings, vol. 81. CEUR-WS.org (2003)
13. Gadducci, F., Heckel, R., Koch, M.: A fully abstract model for graph-interpreted temporal logic. In: Ehrig, H., Engels, G., Kreowski, H.J., Rozenberg, G. (eds.) 6th International Workshop on Theory and Application of Graph Transformations (TAGT'98). LNCS, vol. 1764, pp. 310–322. Springer (2000)
14. Gadducci, F., Lluch Lafuente, A.: Graphical encoding of a spatial logic for the π-calculus. In: Mossakowski, T., Montanari, U., Haveraaen, M. (eds.) 2nd International Conference on Algebra and Coalgebra in Computer Science (CALCO'07). LNCS, vol. 4624, pp. 209–225. Springer (2007)
15. Hazen, A.: Counterpart-theoretic semantics for modal logic. The Journal of Philosophy 76(6), 319–338 (2004)
16. Hodkinson, I.M., Wolter, F., Zakharyaschev, M.: Monodic fragments of first-order temporal logics: 2000-2001 a.d. In: Nieuwenhuis, R., Voronkov, A. (eds.) 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'01). LNCS, vol. 2250, pp. 1–23. Springer (2001)
17. Huth, M., Ryan, M.: Logic in Computer Science: Modelling and Reasoning about Systems (Second Edition). Cambridge University Press (2004)
18. Reif, J., Sistla, A.P.: A multiprocess network logic with temporal and spatial modalities. International Journal of Computer and System Sciences 30(1), 41–53 (1985)
19. Rensink, A.: Towards model checking graph grammars. In: Leuschel, M., Gruner, S., Lo Presti, S. (eds.) 3rd Workshop on Automated Verification of Critical Systems. University of Southampton Technical Reports, vol. DSSE–TR–2003–2, pp. 150–160. University of Southampton (2003)
20. Rensink, A.: Model checking quantified computation tree logic. In: Baier, C., Hermanns, H. (eds.) 17th International Conference on Concurrency Theory (CONCUR'06). LNCS, vol. 4137, pp. 110–125. Springer (2006)
21. Reynolds, J.: Separation logic: A logic for shared mutable data structures. In: 17th IEEE Symposium on Logic in Computer Science (LICS'02). pp. 55–74. IEEE Computer Society (2002)
22. Vandin, A.: Algebraic models for a second-order modal logic. Master's thesis, University of Pisa (2009), http://www.di.unipi.it/∼vandin/thesis.pdf
23. Yahav, E., Reps, T., Sagiv, M., Wilhelm, R.: Verifying temporal heap properties specified via evolution logic. In: Degano, P. (ed.) 12th European Symposium on Programming Languages (ESOP'03). LNCS, vol. 2618, pp. 204–222. Springer (2003)