

Fork Algebras as a Formalism to Reason Across Behavioral Specifications (Extended Abstract)

Marcelo F. Frias^{1*}, Juan P. Galeotti², Carlos G. Lopez Pombo², and Mario Roman²

¹ Department of Computer Science, School of Exact and Natural Sciences, Universidad de Buenos Aires, Pabellón I, Ciudad Universitaria, Buenos Aires, 1428, Argentina, and CONICET. E-mail: mfrias@dc.uba.ar.

² Department of Computer Science, School of Exact and Natural Sciences, Universidad de Buenos Aires, Pabellón I, Ciudad Universitaria, Buenos Aires, 1428, Argentina. E-mail: {jgaleotti, clpombo, mroman}@dc.uba.ar.

Abstract. Describing systems through the specification of different views is a well accepted practice in modern software engineering. In this paper we show how to reason across behavioral specifications within a relational framework. We consider views specifying behavioral information using linear temporal logic or dynamic logic. The main result is that independently generated specifications can be amalgamated within a common relational framework to which different analysis techniques can be applied. The paper also presents a realistic problem for which behavioral specifications in dynamic logic and linear temporal logic are jointly employed in the proof of a non trivial property.

1 Introduction

Describing systems through the specification of its different views is a well accepted practice in modern software engineering. Modeling languages such as UML [1], Syntropy [2], or Catalysis [3], use different views in order to describe different concerns. For instance, it is a common practice to separate static (or structural information) from dynamic (or behavioral) information. In this paper we will concentrate on relating seemingly unrelated behavioral specifications. In order to emphasize this “unrelatedness”, we will assume that views are specified using different behavioral specification formalisms. We could use formalisms such as dynamic logic (propositional or first-order) [4], temporal logic (propositional or first-order) [5], temporal logic of actions [6], or modal action logic [7]. For the sake of simplicity we remain within the propositional realm, although the results presented in this work can be easily extended to a first order setting. This selection was done taking into account that one logic has a state-based semantics, while the other has a trace-based semantics.

* Research partially funded by a grant from Antorchas foundation and project UBA-CYT X094.

Reasoning across different logics can be done in different ways. For instance, for the specific case of dynamic and linear temporal logics, the logic DLTL (Dynamic Linear Temporal Logic) was presented in [8]. Actually, DLTL is not as expressive as both logics put together, but includes significant features from both logics. Nevertheless, it is not our goal to define a mega-logic extending all the formalisms that might be useful towards software specification, but rather to interpret these formalisms into an adequate relational formalism. Another framework that was conceived with the intention of reasoning across logical systems is that of *institutions* [9]. In [10] it is presented a institution based approach to reasoning across dynamic and linear temporal logics. The main concept there, is that of *synchronization*. Essentially, a new logic is defined whose models are pairs of models from each constituent logic. These pairs are constrained by *synchronization rules*. We found that approach too restrictive, in that reasoning across logics requires on the logics engineer a complete grasping of the way both logics get interwoven. In [11], work related to ours is presented, where formulas from one logic are translated to formulas from the other under a certain set of *locality conditions* that are required in order to guarantee that deductions from one formalism are preserved in the other. Usually one logic is recognized as more expressive and is therefore the target to which the other logic gets translated. This is then extended in a way that a resulting formalism is built, to which the original logics are translated. In our approach we use a common unifying formalism (an extension of the relational calculus [12]) as the target formalism. Locality conditions are often captured by simple equations added at the relational level. The translations are semantics preserving mappings from logical formulas to relational terms.

Our approach is based on existing work on relational algebraization of logics [13]. A relational algebraization of a logic \mathcal{L} consists on a semantics-preserving mapping $T_{\mathcal{L}} : \text{Formulas}_{\mathcal{L}} \rightarrow \text{RelTerms}$ mapping \mathcal{L} -formulas to relational terms. Relation algebras are extensions of Boolean algebras and, as such, they have a largest (according to the Boolean ordering) element $\mathbf{1}$. Then, semantics-preservation is characterized by the following condition:

$$\text{for all } \Gamma \cup \{\alpha\} \subseteq \text{Formulas}_{\mathcal{L}},$$

$$\Gamma \models_{\mathcal{L}} \alpha \iff \{T_{\mathcal{L}}(\gamma) = \mathbf{1} : \gamma \in \Gamma\} \vdash_{\text{Eq}} T_{\mathcal{L}}(\alpha) = \mathbf{1} .$$

Several logics have already been relationally algebraized to different extensions of the relational calculus. Among these, Dynamic logic (both propositional and first-order) [14,15], linear temporal logic (both propositional and first-order) [16,17], and classical first-order logic with equality [18] are among the logical systems studied already.

This paper contains two main contributions. First, we show how to reason across dynamic and linear temporal specifications with no need of mixing those concepts that underlay each logic. We prove that every property expressible in the logic DLTL that semantically follows from dynamic specification T_1 and linear temporal specification T_2 , can be equationally proved from the translations

of Γ_1 and Γ_2 . Second, we present a realistic case-study of a system whose atomic actions are specified in dynamic logic, and whose temporal constraints are specified in linear temporal logic. We then prove that the system behaves properly under all possible executions.

2 The Specification Formalisms

Formal specification languages usually mix a logic (from which syntax and semantics for axioms is derived) with different structuring mechanisms that allow a requirements engineer to build more complex specifications from simpler ones. Since adopting particular structuring mechanisms would demand tailoring our results to these, we will assume that a specification is just a theory presentation in a given logic, i.e., a signature presenting what are the extralogical symbols that will be used in the specification, as well as a finite set of axioms describing their meaning.

In this paper we will deal with three logics, namely, propositional dynamic logic, linear temporal logic, and dynamic linear temporal logic. Although all the developments carried on in this paper are valid for first-order versions of these logics, in order to simplify the presentation we will stick to the propositional case.

Propositional Dynamic Logic: The syntax and semantics of propositional dynamic logic (PDL for short) can be found in [4]. PDL is a formalism for reasoning about actions (that can be interpreted as *programs*).

A PDL signature is a pair $\langle A, P \rangle$ where $A = \{a_i\}_{i \in \mathcal{A}}$ is a set of action symbols (atomic programs), and $P = \{p_i\}_{i \in \mathcal{P}}$ are the atomic propositions.

The semantics of PDL is defined on a structure $\mathfrak{K} = \langle S, m_{\mathfrak{K}} \rangle$ where S is a set of elements called *states* and $m_{\mathfrak{K}}$ is a *meaning function* assigning a subset of S to each atomic proposition and a binary relation on S to each program.

Linear Temporal Logic: The linear temporal logic LTL [19] is defined over a set of atomic propositions $P = \{p_i\}_{i \in \mathcal{P}}$.

The semantics of LTL formulas is also defined over a Kripke structure K of the form $\langle S, T, S_0, P, L \rangle$, where S is the set of states, $T \subseteq S \times S$ is the transition relation, $S_0 \subseteq S$ is the set of initial states, P is the set of atomic propositions, and $L : S \rightarrow \mathcal{P}(P)$ is the labelling function that maps every state to a subset of atomic propositions. The transition relation T is assumed to be complete; that is, every state has at least one successor.

Given a Kripke structure K , the set of paths of K is denoted Δ_K . A path $s \in \Delta_K$ is a infinite sequence s_0, s_1, \dots such that $s_i \in S$ and $\langle s_i, s_{i+1} \rangle \in T$ for all $i \geq 0$. We denote by s^i the suffix of s starting at position i . Similarly, we denote by s_i the i -th state in the path s .

Dynamic Linear Time Temporal Logic: Dynamic linear time temporal logic (DLTL for short) [8] is an extension of LTL. The basic idea is to strengthen the until modality by indexing it with PDL test free programs.

Given a signature $\Sigma = \langle A, P \rangle$, where $A = \{a_i\}_{i \in \mathcal{A}}$ is a set of atomic action symbols and $P = \{p_i\}_{i \in \mathcal{P}}$ is a set of proposition symbols. The set of programs of DTL is denoted as *PrgDTL*.

DTL semantics is defined over a structure $K = \langle S, \{A_i\}_{i \in \mathcal{A}}, S_0, P, L \rangle$ where the family of binary relations $A_i \subseteq S \times S$ gives semantics to atomic actions, and $K^{LTL} = \langle S, \bigcup_{i \in \mathcal{A}} A_i, S_0, P, L \rangle$ is a Kripke structure for LTL. In the same way, we define Δ_K with $T = \bigcup_{i \in \mathcal{A}} A_i$.

3 Omega Closure Fork Algebras

Fork algebras [20] are described through a few equational axioms. The intended models of these axioms are structures called *proper fork algebras*, in which the domain is a set of binary relations (on some base set, let us say B), closed under union, intersection, complement, transposition, composition and fork (denoted as ∇); and containing the constants \emptyset (the empty relation), $\mathbf{1}$ (the universal relation on B) and Id (the identity relation on B).

Finally, the binary operation *fork* requires the base set B to be closed under a injective function \star . This means that there are elements x in B that are the result of applying the function \star to elements y and z . Since \star is injective, x can be seen as an encoding of the pair $\langle y, z \rangle$. The application of fork to binary relations R and S is denoted by $R\nabla S$, and its definition is given by:

$$R\nabla S = \{ \langle a, b \star c \rangle : \langle a, b \rangle \in R \text{ and } \langle a, c \rangle \in S \} .$$

Closure fork algebras are then obtained from fork algebras by adding *reflexive-transitive closure*, which, for a binary relation r , is denoted by r^* .

A calculus characterizing the class of proper closure fork algebras can be found in [15]. From now on we will refer to this calculus as FRL.

It is relatively straightforward to prove that proper closure fork algebras are among the models of FRL, since they satisfy the axioms. On the other hand, as it was proved in [15] (which heavily relies on [21]), if a model is not a proper closure fork algebra then it is isomorphic to one.

4 Interpretability Results

In [22], an extension of FRL is defined to interpret DTL logic into fork algebras. In the next sections, we will use this extension (namely FRL^+) as an amalgamating formalism to reason across the PDL, LTL and DTL.

We define FRL^+ by adding the constants \mathbf{S} (states), $\{\mathbf{A}_i\}_{i \in \mathcal{A}}$ (atomic actions), $\{\mathbf{P}_i\}_{i \in \mathcal{P}}$ (propositions), \mathbf{tr} (traces), \mathbf{T} (transition relation) and \mathbf{S}_0 (initial states) and axioms characterizing their behaviour.

Semantics preserving translations from PDL and LTL formulas to fork algebra terms were presented in [14] and [16] respectively.

The translation T_{DLTL} [22] mapping formulas from DLTl to fork algebra terms, is defined inductively as follows:

$$\begin{aligned} T_{DLTL}(p_i) &= \pi; \mathbf{P}_i \quad , \text{ for all } i \in \mathcal{P} \\ T_{DLTL}(\neg\alpha) &= \mathbf{tr}; \overline{T_{DLTL}(\alpha)} \\ T_{DLTL}(\alpha \wedge \beta) &= T_{DLTL}(\alpha) \cup T_{DLTL}(\beta) \\ T_{DLTL}(\alpha \mathbf{U}^P \beta) &= M_{DLTL}(\alpha, P); T_{DLTL}(\beta) \end{aligned}$$

$$\begin{aligned} M_{DLTL}(\alpha, a_i) &= Dom(T_{DLTL}(\alpha)); Ran(\pi \nabla(\mathbf{A}_i \otimes \rho)); \rho \quad , \text{ for all } i \in \mathcal{A} \\ M_{DLTL}(\alpha, R^*) &= (M_{DLTL}(\alpha, R))^* \\ M_{DLTL}(\alpha, R \cup S) &= M_{DLTL}(\alpha, R) \cup M_{DLTL}(\alpha, S) \\ M_{DLTL}(\alpha, R; S) &= M_{DLTL}(\alpha, R); M_{DLTL}(\alpha, S) \end{aligned}$$

Theorem 41 ([22] DLTl Interpretability) *For all $\Gamma \cup \{\alpha\} \subseteq ForDLTL$, we have that,*

$$\Gamma \models_{DLTL} \alpha \quad \text{iff} \\ \{ \mathbf{tr}_0; T_{DLTL}(\gamma) = \mathbf{tr}_0; \mathbf{1} : \gamma \in \Gamma \} \vdash_{FRL^+} \mathbf{tr}_0; T_{DLTL}(\alpha) = \mathbf{tr}_0; \mathbf{1} .$$

5 Reasoning Across Behavioral Specifications: The General Result

As we did with LTL, any DLTl Kripke structure can be seen as a PDL model. Let $K = \langle S, \{A_i\}_{i \in \mathcal{A}}, S_0, P, L \rangle$ be a structure of DLTl, we define K^{PDL} as the structure:

$$K^{PDL} = \langle S, m \rangle$$

where $m(a_i) = A_i$ for all $i \in \mathcal{A}$, and $m(p_i) = L(p_i)$ for all $i \in \mathcal{P}$.

Theorem 51 *Let $\Gamma_1 \subseteq ForPDL$, $\Gamma_2 \subseteq ForLTL$ and $\gamma \in ForDLTL$. Then, for all DLTl Kripke structure K :*

$$\text{If } K^{PDL} \models_{PDL} \Gamma_1 \text{ and } K^{LTL} \models_{LTL} \Gamma_2 \text{ , then } K \models_{DLTL} \gamma$$

is equivalent to the following equational reasoning:

$$\begin{aligned} & \mathbf{S}; T_{PDL}(\alpha) = \mathbf{S}; \mathbf{1}, \text{ for all } \alpha \in \Gamma_1 \\ & \mathbf{tr}_0; T_{LTL}(\beta) = \mathbf{tr}_0; \mathbf{1}, \text{ for all } \beta \in \Gamma_2 \\ & \vdash_{FRL^+} \\ & \mathbf{tr}_0; T_{DLTL}(\gamma) = \mathbf{tr}_0; \mathbf{1} \end{aligned}$$

6 Reasoning Across Behavioral Specifications: The Case-Study

In this section we will present a case-study of verification of properties using FRL^+ . This case-study shows a mobile system where a user interacts with it while on the road. Think for instance of a tourist with his PDA¹ in a city, interacting with a server through a wireless connection.

Both user and server can trigger *actions* (i.e.: requesting information, serving enqueued requests, etc.). Actions might lead to a transition between different system states. Such behaviour can be formalized by stating a precondition and a postcondition for each atomic action using PDL formulas of the form:

$$pre_i \Longrightarrow [act_i]post_i$$

where pre_i and $post_i$ are the precondition and postcondition of the action act_i .

Then we would like to capture the fact that the validity of certain linear temporal property I is preserved by execution of every atomic action. This behaviour is stated by formulas of the form:

$$\Box((pre_i \wedge I) \Longrightarrow \oplus(post_i \Longrightarrow I))$$

again, considering that pre_i and $post_i$ are the precondition and postcondition of the action act_i .

Finally, we could like to prove that every test free program in the regular language generated by the set of atomic actions preserves the validity of the linear temporal property I . In other words, we would like to prove that for all R test free program the following formula holds:

$$(\mathbf{true} \cup^R \mathbf{true}) \Longrightarrow (I \Longrightarrow I \cup^R I)$$

7 Conclusions and Further Work

We have presented a interpretability result for the dynamic linear temporal logic (DLTL). Also, we proposed a relational framework (FRL^+) to reason across PDL, LTL and DLTL, and proved that it is equivalent to reason within the semantics of the logics and the proposed relational calculus.

We use this framework to verify a non trivial property that combines both dynamic and linear temporal concepts. Finally, we present a realistic problem in which such reasoning is relevant.

References

1. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison–Wesley Longman Publishing Co., Inc. (1998)

¹ Personal digital assistant, a handheld device that combines computing, telephone/fax, Internet and networking features.

2. Cook, S., Daniels, J.: Designing object systems: Object-oriented modelling with Syntropy. Prentice Hall (1994)
3. D'Souza, D.F., Wills, A.C.: Objects, components, and frameworks with UML: The Catalysis(SM) approach. Object Technology Series. Addison-Wesley Longman Publishing Co., Inc. (1998)
4. Harel, D., Kozen, D., Tiuryn, J.: Dynamic logic. Foundations of Computing. MIT Press (2000)
5. Manna, Z., Pnueli, A.: Temporal Verification of Reactive Systems. Springer-Verlag, New York (1995)
6. Lammport, L.: Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley Longman Publishing Co., Inc. (2002)
7. Jeremaes, P., Khosla, S., Maibaum, T.S.E.: A modal (action) logic for requirements specification. In Barnes, D., Brown, P., eds.: Proceedings of IEE Software Engineering. Volume 86 of IEE Computer Series 6., Petro Peregrinis Ltd. (1986) 278–294
8. Henriksen, J.G., Thiagarajan, P.: Dynamic linear time temporal logic. Annals of Pure and Applied Logic **96** (1999) 187–207
9. Goguen, J.A., Burstall, R.M.: Introducing institutions. In Clarke, E.M., Kozen, D., eds.: Proceedings of the Carnegie Mellon Workshop on Logic of Programs. Lecture Notes in Computer Science, Springer-Verlag (1984) 221–256
10. Sernadas, A., Sernadas, C., Caleiro, C.: Synchronization of logics with mixed rules: Completeness preservation. In Johnson, M., ed.: Proceedings of the 1997 Algebraic Methodology and Software Technology – AMAST 97. Lecture Notes in Computer Science, Macquarie University, Sydney, Australia, Springer-Verlag (1997) 465–478
11. Dimitrakos, T., Bicarregui, J., Maibaum, T.S.E.: Integrating heterogeneous formalisms: Framework and application. Manuscript (1999)
12. Tarski, A.: On the calculus of relations. Journal of Symbolic Logic **6** (1941) 73–89
13. Henkin, L.A., Monk, J., Tarski, A.: Cylindric algebras, part II. Volume 115. North Holland (1985)
14. Frias, M.F., Orłowska, E.: Equational reasoning in non-classical logics. Journal of Applied Non-classical Logics **8** (1998) 27–66
15. Frias, M.F., Baum, G.A., Maibaum, T.S.E.: Interpretability of first-order dynamic logic in a relational calculus. In de Swart, H., ed.: Proceedings of the 6th. Conference on Relational Methods in Computer Science (RelMiCS) - TARSKI, Oisterwijk, The Netherlands (2001) 66–80
16. Frias, M.F., Pombo, C.G.L.: Time is on my side. In Berghammer, R., Möller, B., eds.: Proceedings of the 7th. Conference on Relational Methods in Computer Science (RelMiCS) - 2nd. International Workshop on Applications of Kleene Algebra, Malente, Germany (2003) 105–111
17. Frias, M.F., Pombo, C.G.L.: Interpretability of first-order linear temporal logics in fork algebras. Journal of Logic and Algebraic Programming (2004) in press.
18. Frias, M.F.: Fork algebras in algebra, logic and computer science. Logic and Computer Science. World Scientific Publishing Co. (2002)
19. Emerson, E.A.: Temporal and modal logic. In van Leeuwen, J., ed.: Handbook of Theoretical Computer Science. Volume B. Elsevier, Amsterdam (1990)
20. Haeberer, A.M., Veloso, P.A.S.: Partial relations for program derivation: adequacy, inevitability and expressiveness. In: Proceedings of the Working Conference on Constructing Programs from Specifications '91, IFIP TC2, Constructing Programs from Specifications, North Holland (1991) 310–352
21. Frias, M.F., Haeberer, A.M., Veloso, P.A.S.: A finite axiomatization for fork algebras. Logic Journal of the IGPL **5** (1997) 311–319

22. Galeotti, J.P., Roman, M.: Reasoning across dynamic logic and linear temporal logic using fork algebras. Master's thesis, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires (2004) Advisors: Marcelo F. Frias and Carlos G. Lopez Pombo.