Induction in First-Order Logic with Temporal Metric Operators

Pedro Torres and Marian Ursu

Narrative and Interactive Media Group
Department of Computing
Goldsmiths, University of London
{p.torres,m.ursu}@gold.ac.uk

Abstract. A preliminary exploration of induction in the realm of first-order logic enriched with temporal metric operators is presented. The Interaction Modelling Language (Imola) is formally defined, it is shown to have a lattice structure with respect to an extended subsumption relation and locally finite and complete refinement operators are introduced.

Keywords: temporal logic, metric operators, induction, social communication

1 Introduction

Automated Orchestration [14, 2] concerns making automatic camera decisions in a multi-location multi-camera mediated video communication, relying on cues automatically extracted in real time. Examples of such cues include face position, voice activity, direction of gaze, focus of attention and keyword usage. First-order logic seems suitable to describe background knowledge in this communication scenario but it lacks the ability to express, in a compact way, particular sequences of events and time intervals between them; such sequences of events constitute the basic building blocks of most orchestration rules (c. f. [2]). In this paper, we consider bringing metric temporal operators into first-order logic — to obtain the *Interaction Modelling Language* (Imola) — and study its inductive properties.

Temporal Logic is a framework for reasoning about the temporal evolution of properties of a system which has been the subject of intensive research in the past decades and has found numerous applications in computer science. Temporal Logic based on the two modalities *since* and *until* has been proved to be equivalent in terms of expressiveness to monadic first-order logic [4]. However, various incompleteness results [8, 13] led to the conclusion that many useful temporal logics were not even recursively enumerable. Despite the negative results, restrictions on the number of variables within temporal operators were shown to yield decidable fragments of temporal logic [3] suggesting that carefully constraining the syntax of the language could be used to contain complexity issues.

Traditional approaches to temporal logic are qualitative with respect to time in that there is no explicit handling of time differences between events. For automated orchestration, a quantitative approach seems required. Although the introduction of temporal metric operators in qualitative temporal logics often leads to a drastic increase in complexity of satisfiability [11] — typically from NP or PSPACE completeness, to EXPSPACE-completeness or even undecidability — it has been shown that the temporal logic obtained by extending *since/until* continuous temporal logic with the operator "sometime within n time units in the future" is still PSPACE-complete [7].

With this in mind, we have chosen metric operators which, on the one hand, provide sufficient expressive power for orchestration and, on the other hand, have been shown not to bring major complexity issues. To further minimise complexity, we restrict the usage of temporal operators to the front of literals. Imola has been designed precisely with the goal of making orchestration rules and associated background knowledge simple to express, while at the same time both keeping tractability and maintaining a general purpose flavour. In this preliminary exploration, we restrict our metric operators to "sometime within n time units", $\lhd_n$, "sometime later than n time units from now, $\rhd_n$, and "in exactly n time units", $\odot_n$, but further extensions could be considered to encompass other temporal relations such as those considered in [1].

Inductive properties of temporal logic have been studied in [5], where an extension of a Prolog-style language which allows temporal operators such as *since* and *until* in front of literals is identified and that approach is followed closely here.

The structure of the paper is as follows. Firstly, Imola is formally defined via its syntax (§2.1) and declarative semantics (§2.2). Secondly, it is shown that the natural extension of first-order subsumption to Imola provides the set of clauses with a lattice structure (§3.1), as proved by the existence of greatest specialisation and least generalisation under subsumption for arbitrary pairs of clauses. Additionally, both downward and upward refinement operators are defined for Imola and they are shown to be locally finite and complete (§3.2). The last section (§4) concludes and points directions of future work.

2 The IMOLA language

Analogously to other forms of temporal logic, Imola concerns reasoning about time-dependent properties. However, in contrast with standard temporal logic, Imola can make use of an explicit metric on time. Imola can express statements such as "A happened and at most 5 time units after that B happened", using explicit temporal differences between events. The reason for the introduction of such concrete manipulation of time stems from the particular application Imola was designed for, automated orchestration, which was introduced in the previous section.

2.1 Syntax

A *signature* Σ is a tuple $(\mathcal{X}, \mathcal{F}, \mathcal{P}, \alpha)$, where $\mathcal{X}$ is a countable set of *variable symbols*, $\mathcal{F}$ and $\mathcal{P}$ are, respectively, finite sets of *function symbols* and *predicate* *symbols*, and $\alpha : \mathcal{F} \cup \mathcal{P} \to \mathbb{N}$ is the *arity function.* For the remainder of this section, let $\Sigma = (\mathcal{X}, \mathcal{F}, \mathcal{P}, \alpha)$ be an arbitrary but fixed signature.

The set of IMOLA *terms over signature* Σ, denoted $\mathcal{T}(\Sigma)$, is inductively defined by imposing that (i) each $x \in \mathcal{X}$ be a term and that (ii) each $f(t_1, \ldots, t_{\alpha(f)})$ be a term, provided $\forall_i . t_i \in \mathcal{T}(\Sigma)$.

The set of *atoms over signature* Σ, denoted $\mathcal{A}(\Sigma)$, is inductively defined as the smallest set of objects closed under the following rules: (i) $\top$ and $\bot$ are atoms; (ii) each $P(t_1, \ldots, t_{\alpha(P)})$ is an atom, provided $\forall_i . t_i \in \mathcal{T}(\Sigma)$; (iii) if φ is an atom, then, for any $n \in \mathbb{N}$, $\varphi \uparrow$, $\varphi \downarrow$ and $\odot_n \varphi$ are atoms; (iv) if φ and ϕ are atoms, then, for any $n \in \mathbb{N}$, $\varphi \lhd_n \phi$ and $\varphi \rhd_n \phi$ are atoms. The operators in rules (iii) and (iv) will be called *temporal operators.*

The set of *literals over signature* Σ, denoted $\mathcal{L}(\Sigma)$ is inductively defined by the same four rules as the set of atoms with the additional rule that (v) if φ is a literal then so is $\neg\varphi$.

Formulas are defined inductively as the smallest set of objects closed under the following rules: (i) if φ is a literal then it is a formula; (ii) if φ_1 and φ_2 are formulas, then so are $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \to \varphi_2$, and $\varphi_1 \leftrightarrow \varphi_2$; (iii) if φ is a formula and $x \in \mathcal{X}$ then both $\forall x.\varphi$ and $\exists x.\varphi$ are formulas. The IMOLA *language* is defined as the set of all formulas.

A *fact over signature* Σ in IMOLA is simply an atom $\varphi \in \mathcal{A}(\Sigma)$ and is usually written as "φ.". A *rule over signature* Σ is either a fact over Σ or a pair $(\varphi, \{\varphi_1, \ldots \varphi_n\})$, usually written "$\varphi \ :- \ \varphi_1, \ldots, \varphi_n$." where φ is an atom, called the *head* of the rule, and each φ_i is a literal, with the set $\{\varphi_1, \ldots \varphi_n\}$ being called the *tail* of the rule. In rules, negated atoms are written $\texttt{not}(\varphi)$ which means 'negation as failure' and not logical negation, as usual in logic programming. Note that the tail may contain literals which are not atoms, contrary to the usual practice in Horn languages. Rules may be denoted simply by a set of literals. An IMOLA *program* is defined as a finite set of rules over Σ.

A *substitution* is a mapping $\sigma : \mathcal{X} \to \mathcal{T}(\Sigma)$ for which only a finite number of domain elements are not fixed points. Two terms $t_1, t_2 \in \mathcal{T}(\Sigma)$ are *unifiable* if there is a substitution σ, called a *unifier*, s.t. $\sigma(t_1) = \sigma(t_2)$. The unifier is called a *most general unifier*, denoted $\sigma = \mathrm{mgu}(t_1, t_2)$, if, for every unifier σ_1, there is a substitution σ_2 s.t. $\sigma = \sigma_1 \circ \sigma_2$, where composition of substitution is defined as usual. These definitions carry over to rules in the natural way. Robinson's unification theorem [12] extends to IMOLA and if two terms or rules are unifiable, then there exists a unique mgu up to renaming of variables.

2.2 Declarative Semantics

The declarative semantics is defined through the logical consequence relation, $\models$. For the remainder of this section, let $\Sigma = (\mathcal{X}, \mathcal{F}, \mathcal{P}, \alpha)$ be an arbitrary but fixed signature. Following the *possible world semantics* approach [6], an *interpretation* is defined as a tuple

$$\mathcal{J} = (U_{\mathcal{J}}, S, s_0, \delta_1, \delta_2, w, \mathcal{I})$$

where $U_{\mathcal{J}}$ is the *universe* of the interpretation, S is a set of *states*, s_0 is the *inital state* and is required to be an element of S, the $\delta_i \subseteq S \times S$ are *accessibility* *relations* (δ_1 is the next state relation and δ_2 its transitive closure), $w : \mathcal{X} \to U_{\mathcal{J}}$ is an *evaluation*, and $\mathcal{I}$ is a function which associates a standard first-order interpretation to each state.

Terms are evaluated in the natural way, by imposing $\mathcal{J}(t) = w(t)$ when t is just a variable and $\mathcal{J}(t) = \mathcal{I}(s_0)(f)\ (\mathcal{J}(t_1), \ldots, \mathcal{J}(t_n))$ for $t = f(t_1, \ldots, t_n)$. To interpret a formula and give it meaning, the usual rules go for standard non-temporal literals and the following rules are added to deal with the others:

$$\begin{array}{l}
\mathcal{J}(\varphi \rhd_n \phi) = 1 \text{ iff } s_0\delta_1^m s,\ \mathcal{J}[s](\phi) = 1,\ \mathcal{J}[s_0](\varphi) = 1 \text{ and } m > n, \text{ for some } s \in S \\
\mathcal{J}(\varphi \lhd_n \phi) = 1 \text{ iff } s_0\delta_1^m s,\ \mathcal{J}[s](\phi) = 1,\ \mathcal{J}[s_0](\varphi) = 1 \text{ and } m < n, \text{ for some } s \in S \\
\mathcal{J}(\odot_n \phi) = 1 \quad \text{iff } s_0\delta_1^n s \text{ and } \mathcal{J}[s](\phi) = 1
\end{array}$$

where $\mathcal{J}[s](\phi)$ stands for the interpretation obtained by setting the initial state of $\mathcal{J}$ to s.

An interpretation $\mathcal{J}$ is called a *model* of a formula φ, denoted $\mathcal{J} \models \varphi$, if $\mathcal{J}(\varphi) = 1$. As usual, an IMOLA formula is *valid*, *satisfiable* or *unsatifiable*, if, respectively, all interpretations are models of φ, there is an interpretation which is a model of φ, or there is no interpretation which is a model of φ.

3 Induction in IMOLA

3.1 Lattice structure induced by subsumption

Let φ_1 and φ_2 be IMOLA literals other than $\bot$ and $\top$. As with standard subsumption, φ_1 *subsumes* φ_2, denoted $\varphi_1 \succeq \varphi_2$, iff there is a substitution θ such that $\theta(\varphi_1) = \varphi_2$. For the excluded cases involving $\bot$ and $\top$, define $\bot \succeq \varphi$ and $\varphi \succeq \top$, for every literal φ.

Theorem 1. *The set of literals $\mathcal{L}$ endowed with the quasi order $\succeq$ forms a lattice.*

Proof. It is sufficient to show that given two literals φ_1 and φ_2 there exist a greatest specialisation and a least generalisation in $\mathcal{L}$.

Regarding specialisation, define a function $s : \mathcal{L} \times \mathcal{L} \to \mathcal{L}$ such that: (i) if φ_1 and φ_2 are unifiable with $\sigma = \mathrm{mgu}(\varphi_1, \varphi_2)$, then $s(\varphi_1, \varphi_2) = \sigma(\varphi_1)$ and (ii) if they are not unifiable, then $s(\varphi_1, \varphi_2) = \texttt{true}$. Function s is indeed a greatest specialisation. Details are left out.

Regarding generalisation, define a function $g : \mathcal{L} \times \mathcal{L} \to \mathcal{L}$ such that: (i) if φ_1 is an atom and φ_2 is a negated atom or vice-versa, then $g(\varphi_1, \varphi_2) = \texttt{true}$; (ii) if φ_1 and φ_2 are negated atoms[1], with $\varphi_1 = \texttt{not}(\phi_1)$ and $\varphi_2 = \texttt{not}(\phi_2)$, then $g(\varphi_1, \varphi_2) = \texttt{not}\ g(\phi_1, \phi_2)$; (iii) if φ_1 and φ_2 are both atoms, there are three distinct classes of atoms — predicates, unary temporal operators, binary temporal operators.

For predicates with different symbols, define $g(\varphi_1, \varphi_2) = \texttt{false}$; for predicate with equal symbols, $P(t_1, \ldots, t_n)$ and $P(\tau_1, \ldots, \tau_n)$, define $g(\varphi_1, \varphi_2) =$

[1] The lattice structure under analysis is for literals of the Horn-like language with negation as failure, hence negation of φ is written $\texttt{not}(\varphi)$.

$P(\mathrm{LGS}(t_1, \tau_1), \ldots, \mathrm{LGS}(t_n, \tau_n))$ where $\mathrm{LGS}(t_1, t_2)$ can be computed using the usual anti-unification algorithm [10]. For unary temporal operators, define, for instance, $g(\phi_1 \uparrow, \phi_2 \uparrow) = (g(\phi_1, \phi_2)) \uparrow$, and similarly for the operators $\downarrow$ and $\odot_n$. For binary temporal operators, define, for instance, $g(\phi_1 \lhd_n \psi_1, \phi_2 \lhd_n \psi_2) = (g(\phi_1, \phi_2)) \lhd_n (g(\psi_1, \psi_2))$, and similarly for the operator $\rhd_n$. Function g is indeed a least generalisation. Details are left out. $\square$

As in first-order logic, a rule R_1 is said to *subsume* a rule R_2, denoted $R_1 \succeq R_2$, if there exists a substitution θ such that $\theta(R_1) \subseteq \theta(R_2)$, where the rules are being represented as sets of literals. This gives rise to the following theorem presented without proof.

Theorem 2. *The set of* Imola *rules endowed with the quasi order $\succeq$ forms a lattice.*

3.2 Refinement Operators in Imola

Let $\Sigma = (\mathcal{X}, \mathcal{F}, \mathcal{P}, \alpha)$ be a signature and $\phi \in \mathcal{L}(\Sigma)$. Downward and upward refinement operators for rules will now be only sketched due to space constraints.

The downward refinement operator $\rho_d : 2^{\mathcal{L}(\Sigma)} \to 2^{2^{\mathcal{L}(\Sigma)}}$ is defined as follows. Let R be an Imola rule. It can be represented by a set of literals and R will be overloaded to denote that set in the sequel. Define ρ_d as by the standard mapping within first-order logic (c. f. Definition 17.14 in [9]) but extend it to temporal operators by adding the rules: (i) if $\varphi \in R$ then add $\odot_n \varphi$ and $\mathtt{not}(\odot_n \varphi)$, for every n, to the range of ρ_d; and similarly for all other unary temporal operators; (ii) if $\varphi_1, \varphi_2 \in R$ then add $\varphi_1 \lhd_n \varphi_2$ and $\mathtt{not}(\varphi_1 \lhd_n \varphi_2)$, for every n, to the range of ρ_d; and similarly for all other binary temporal operators.

The upward refinement operator $\rho_u : 2^{\mathcal{L}(\Sigma)} \to 2^{2^{\mathcal{L}(\Sigma)}}$ is defined similarly by the natural extension of the standard mapping within first-order logic (c. f. Definition 17.20 in [9]) and is ommitted for brevity.

Theorem 3. *Both ρ_d and ρ_u refinement operators for* Imola *rules as defined above are locally finite, complete but not proper.*

Proof. Local finiteness is straightforward from both definitions as the number of possible substitutions is finite. Regarding completeness, the standard first-order proof extends to Imola since every most general literal is added to the set of refinements. Details are left out. Absence of properness is directly deduced from the fact that both refinement operators are locally finite and complete, and that there are no ideal refinement operators in clausal languages with at least one predicate or function symbol of arity greater or equal to 2. $\square$

4 Conclusions and Future Work

A novel first-order temporal language, Imola, is introduced, allowing explicit time-metric statements such as “John spoke and less than 5 time units after that, Mary spoke", written Talk(`john`) $\lhd_5$ Talk(`mary`). Syntax and declarative semantics for IMOLA are given. Some inductive properties of the language are studied as well, and it is concluded that the set of IMOLA rules is given a lattice structure by a natural extension of subsumption. Finally, upward and downward refinement operators are defined and shown to be locally finite and complete.

Future work includes: (i) establishing IMOLA tableaux proof procedures which are of practical use; (ii) considering efficient refinement strategies to discard variants of rules and improving search by making explicit use of the fact that time is totally ordered (and therefore $\varphi \lhd_4 \phi$ implies $\varphi \lhd_5 \phi$); (iii) study how extending IMOLA's expressive power (by allowing, for example, statements such as "A happened and at most 5 time units after that B happened, with C holding between the two events") affects tractability of proof procedures and PAC-learnability.

References

1. J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.
2. M. Falelakis, R. Kaiser, W. Weiss, and M. Ursu. Reasoning for Video-mediated Group Communication. In *Proceedings of the IEEE International Conference on Multimedia & Expo*, Barcelona, Spain, 2011.
3. I. Hodkinson, F. Wolter, and M. Zakharyaschev. Decidable fragments of first-order temporal logics. *Annals of Pure and Applied Logic*, 106:85–134, 2000.
4. H. Kamp. *Tense Logic and the Theory of Linear Order.* PhD thesis, UCLA, 1968.
5. R. Kolter. *Inductive Temporal Logic Programming.* PhD thesis, University of Kaiserlautern, 2009.
6. S. Kripke. Semantical Analysis of Modal Logic. *Zeitschrift fur Mathematische Logic und Grundlagen der Mathematik*, 9:67–96, 1963.
7. C. Lutz, D. Walther, and F. Wolter. Quantitative temporal logics over the reals: PSpace and below. *Information and Computation/Information and Control*, 205:99–123, 2007.
8. S. Merz. Decidability and Incompleteness Results for First-Order Temporal Logics of Linear Time. *Journal of Applied Non-classical Logic*, 2, 1992.
9. S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming.* Number 1228 in Lecture Notes in Computer Science. Springer, 1997.
10. G. D. Plotkin. A Note on Inductive Generalization. In *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, 1970.
11. M. Reynolds. The Complexity of Temporal Logic over the Reals. *Annals of Pure and Applied Logic*, 161:1063–1096, 1999.
12. J. A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.
13. A. Szalas and L. Holenderski. Incompleteness of First-Order Temporal Logic with Until. *Journal of Theoretical Computer Science*, 57:317–325, 1988.
14. M. Ursu, P. Torres, V. Zsombori, M. Frantzis, and R. Kaiser. Entertaining each other from a Distance: Orchestrating Video Communication and Play. In *Proceedings of the ACM Multimedia Conference*, Scottsdale (AZ), USA, 2011.