LTL with a sub-order

Amaldev Manuel
amal@imsc.res.in

Institute of Mathematical Sciences
C.I.T Campus, Taramani, Chennai, India-600113

Abstract. By a small parable about extending LTL with a sub-order, we try to describe the moral of Data language paradigm.

1 Introduction

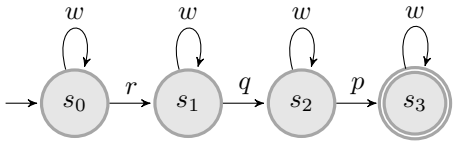
LTL (propositional discrete linear time temporal logic) [Pnueli, 1977] formulas work on histories[i] of the form

$$w = w_1 w_2 \dots w_n, w_i \in P$$

where P is the set of all propositions and is evaluated with respect to a particular time point $i \in |w|$[ii]. The commonly used temporal connectives are $\oplus$ for FUTURE, $\ominus$ for PAST, $\oplus$ for TOMORROW, $\ominus$ for YESTERDAY, u for UNTIL and s for SINCE. The duals of $\oplus$ and $\ominus$ are $\boxplus$ for HENCEFORTH and $\boxminus$ for HITHERTO.

It is well-known that LTL$(s, u, \oplus, \ominus)$ is expressively complete with respect to FO$(<)$ [Kamp, 1968], whereas LTL$(\oplus, \ominus, \oplus, \ominus)$ and LTL$(\oplus, \ominus)$ are complete with respect to FO$^2(<, \prec)$ and FO$^2(<)$ respectively[Etessami et al., 1997][iii][iv]. The fact that the logic is propositional and the lower complexity of its decision problems[v] makes LTL a suitable specification mechanism for describing sequential behaviours of computing machineries.

A transition system describing a printer system with only one process. The actions are request, wait, queue and print abbreviated here as r, w, q, p respectively.



Every request is eventually followed by a queue and in turn by a print. A sample execution trace of the above described system is given below.

$$w \to w \to r \to w \to w \to q \to w \to w \to p \to w$$

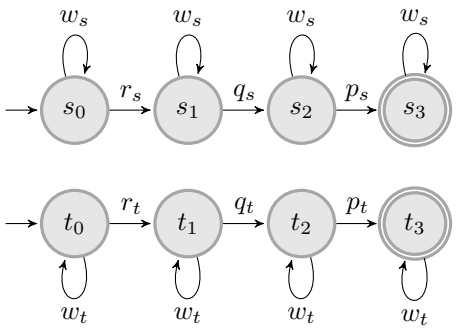
[i] In this paper we consider only LTL over finite histories.
[ii] $|w|$ denote the length of w.
[iii] FO^2 is the two variable fragment of first order logic.
[iv] $\prec$ denotes the successor relation, which is the Hasse covering relation for $<$.
[v] LTL satisfiability problem is in PSPACE whereas FO$(<)$ satisfiability is NON-ELEMENTARY [Stockmeyer, 1974] and FO^2 satisfiability is NEXPTIME-COMPLETE[Etessami et al., 1997].

A desirable property is fairness which states that every request is eventually queued and then printed. Stated in the language of LTL it is as follows.

$$r \to \oplus(q \wedge \oplus p) \wedge \boxplus(r \to \oplus(q \wedge \oplus p))$$

If we have two asynchronous processes, we could still use LTL by adding identity to actions corresponding to each process by means of additional propositional variables, as shown below.

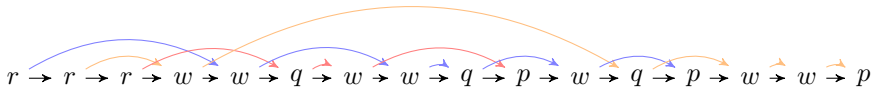
A transition system describing a printer system with two asynchronous processes. The actions request, wait, queue and print are paired with the identity of the process, a boolean value. The invariant that every request is eventually followed by a queue and in turn by a print holds individually in each of the processes. The behaviour of the whole system will be the arbitrary interleaving-s of the behaviours of the individual processes.



The fairness is stated in terms of actions of each processes, in the following way.

$$(r_s \to \oplus(q_s \wedge \oplus p_s) \wedge r_t \to \oplus(q_t \wedge \oplus p_t)) \wedge \boxplus(r_s \to \oplus(q_s \wedge \oplus p_s) \wedge r_t \to \oplus(q_t \wedge \oplus p_t))$$

This way of adding propositional variables for identities does not generalize if the system contains an unbounded number of processes[vi]. In which case, one way to represent the execution trace is by annotating the sequential trace ordered $<$, by yet another order $\lesssim$ as shown below. The following is an execution trace of printer system, the coloured edges denote the execution of each process.



The order $\lesssim$ satisfies the following properties,

1. $\lesssim$ is compatible with $<$, that means, if $i \lesssim j$ implies that $i < j$.
2. The order relation $\lesssim$ is a union of chains, this is by virtue of the fact that each chain in $\lesssim$ is a sequential trace of a process.

Henceforth $<$ denotes a total order and $\lesssim$ stands for a subset of $<$ which satisfies the above properties. One way to specify properties of such structures is to

[vi] Unbounded here means finite but not bounded by a constant.

define temporal connectives which take into account both the order relations[vii]. We define the following temporal logic,

$$\varphi ::= p \mid \oplus_{\sim}\varphi \mid \oplus_{*}\varphi \mid \ominus_{\sim}\varphi \mid \ominus_{*}\varphi \mid \neg\varphi \mid \varphi_1 \vee \varphi_2$$

The semantics of the logic is given with respect to histories ordered by $<$ and $\lesssim$, in the following way, boolean cases are as usual,

$$\begin{array}{lcl} w, i \vDash p & \Leftrightarrow & w_i = p \\ w, i \vDash \oplus_{\sim}\varphi & \Leftrightarrow & \exists j.\ i \lesssim j \wedge w, j \vDash \varphi \\ w, i \vDash \oplus_{*}\varphi & \Leftrightarrow & \exists j.\ i < j \wedge i \not\lesssim j \wedge w, j \vDash \varphi \\ w, i \vDash \ominus_{\sim}\varphi & \Leftrightarrow & \exists j.\ j \lesssim i \wedge w, j \vDash \varphi \\ w, i \vDash \ominus_{*}\varphi & \Leftrightarrow & \exists j.\ j < i \wedge j \not\lesssim i \wedge w, j \vDash \varphi \end{array}$$

We say the history $w \vDash \varphi$ if $w, 1 \vDash \varphi$. We can define $\boxplus_{\sim}\varphi = \neg \oplus_{\sim} \neg\varphi$, $\boxplus_{*}\varphi = \neg \oplus_{*} \neg\varphi$, $\boxplus\varphi = \boxplus_{\sim}\varphi \wedge \boxplus_{*}\varphi$, $\oplus\varphi = \oplus_{\sim}\varphi \vee \oplus_{*}\varphi$. Symmetrically we can define the past modalities as well. The fairness condition can be expressed in this logic as

$$(r \rightarrow \oplus_{\sim}(q \wedge \oplus_{\sim} p)) \wedge \boxplus(r \rightarrow \oplus_{\sim}(q \wedge \oplus_{\sim} p))$$

One benchmark for measuring the expressiveness of temporal logic is of course the classical first order logic. By the standard translation of modal logics we can show that,

Proposition 1. *Every* $\mathrm{LTL}(\oplus_{\sim}, \oplus_{*}, \ominus_{\sim}, \ominus_{*})$ *formula* φ *can be converted to an equivalent* $\mathrm{FO}^2(<, \lesssim)$ *formula* $\hat{\varphi}(x)$*, where* $|\hat{\varphi}(x)| \in \mathcal{O}(|\varphi|)$ *and* $qdp(\hat{\varphi}(x)) = odp(\varphi)$[viii].

Proof. Use the standard translation to obtain $\hat{\varphi}(x)$ from φ, boolean cases are omitted.

$$\begin{array}{ll} ST_x(p) & := p(x) \\ ST_x(\oplus_{\sim}\varphi) & := \exists y.\ x \lesssim y \wedge ST_y(\varphi) \\ ST_x(\oplus_{*}\varphi) & := \exists y.\ x < y \wedge x \not\lesssim y \wedge ST_y(\varphi) \\ ST_x(\ominus_{\sim}\varphi) & := \exists y.\ y \lesssim x \wedge ST_y(\varphi) \\ ST_x(\ominus_{*}\varphi) & := \exists y.\ y < x \wedge y \not\lesssim x \wedge ST_y(\varphi) \end{array}$$

□

The other direction is the interesting one, along the lines of [Etessami et al., 1997] we can show that,

[vii] Linear orders with added relations have been studied recently, from different perspectives. CaRet introduced in [Alur et al., 2004,Alur and Madhusudan, 2006] works over nested words, where the words are ornamented with a relation μ–which is a union of chains of length two, looks at program executions where the relation μ corresponds to call-return patterns which are inherently nested. Another logic one finds in the literature is the LTL $\downarrow$, LTL with freeze quantifiers, introduced in [Demri and Lazic, 2006], which addresses words over an alphabet $\Sigma \times \Delta$ where Σ is a finite alphabet and Δ is an infinite data domain.

[viii] quantifier depth and operator depth respectively.

Theorem 1. *Every* $\mathrm{FO}^2(<,\lesssim)$ *formula* $\varphi(x)$ *can be converted to an equivalent* $LTL(\oplus_\sim, \oplus_\not, \ominus_\sim, \ominus_\not)$ *formula* φ'*, where* $|\varphi'| \in 2^{\mathcal{O}(|\varphi|(qdp(\varphi)+1))}$ *and* $odp(\varphi') = qdp(\varphi)$.

Proof. The proof is by induction on the structure of the formulas. When $\varphi(x)$ is atomic, that is $\varphi = p(x)$, $\varphi' = p_i$. When $\varphi(x)$ is composite, that is $\varphi(x) = \neg\varphi_1(x)$ (or $\varphi(x) = \varphi_1(x) \vee \varphi_2(x)$), we recursively compute $\varphi'_1(x)$ (or $\varphi'_1(x)$ and $\varphi'_2(x)$) and output $\neg\varphi'_1$ (or $\varphi'_1 \vee \varphi'_2$).

The remaining cases are when $\varphi(x)$ is of the form $\exists x.\varphi_1(x)$ or $\exists y.\varphi_1(x,y)$. In the first case φ is equivalent to $\exists y.\varphi_1(y)$ (by renaming) and hence reduces to the second case (considering x as a dummy variable). In the second case we rewrite $\varphi_1(x,y)$ in the form

$$\varphi_1(x,y) = \beta\left(\chi_0(x,y), \ldots, \chi_{r-1}(x,y), \xi_0(x), \ldots, \xi_{s-1}(x), \zeta_0(y), \ldots, \zeta_{t-1}(y)\right)$$

where β is a boolean formula, each χ_i is an order formula, ξ_i is an atomic or existential FO^2 formula with $qdp(\xi_i) < qdp(\varphi)$ and ζ_i is an atomic or existential FO^2 formula with $qdp(\zeta_i) < qdp(\varphi)$. We next pull out the ξ_i's from β by doing a case distinction on which of the sub-formulas ξ_i hold or not. Rewriting the previous expression as,

$$\bigvee_{\bar{\gamma}\in\{\top,\bot\}^s} \left(\bigwedge_{i<s} (\xi_i \leftrightarrow \gamma_i) \wedge \exists y.\beta\left(\chi_0, \ldots, \chi_{r-1}, \gamma_0, \ldots, \gamma_{s-1}, \zeta_0, \ldots, \zeta_{t-1}\right) \right)$$

Next we do a case distinction on which order relation, called *order type*, holds between x and y. All possible relations which can exist between x and y which satisfy the conditions for $\lesssim$ will be an order type, namely, $x = y$, $x \lesssim y$, $x < y \wedge x \not\lesssim y$, $y \lesssim x$, $x > y \wedge y \not\lesssim x$[ix]. When we assume that one of these order types is true, each atomic formula evaluates to either $\bot$ or $\top$ and in particular, each of the ξ's evaluates to either $\bot$ or $\top$; which we denote by ξ^τ. Finally, we can rewrite φ as follows, where Υ stands for the set of all order types:

$$\bigvee_{\bar{\gamma}\in\{\top,\bot\}^s} \left(\bigwedge_{i<s} (\xi_i \leftrightarrow \gamma_i) \wedge \bigwedge_{\tau\in\Upsilon} \exists y\left(\tau \wedge \beta\left(\chi_0^\tau, \ldots, \chi_{r-1}^\tau, \gamma_0, \ldots, \gamma_{s-1}, \zeta_0, \ldots, \zeta_{t-1}\right)\right) \right)$$

If τ is an order type and $\psi(y)$ an FO^2 formula then for $\exists y.\tau \wedge \psi(y)$, an equivalent LTL formula $\tau\langle\psi\rangle$ can be obtained in the following way,

τ	$x = y$	$x \lesssim y$	$x < y \wedge x \not\lesssim y$	$y \lesssim x$	$x > y \wedge y \not\lesssim x$
$\tau\langle\psi\rangle$	ψ	$\oplus_\sim\psi$	$\oplus_\not\psi$	$\ominus_\sim\psi$	$\ominus_\not\psi$

Now, we recursively compute $\xi'_i, i < s$ and $\zeta'_i, i < t$ and outputs,

$$\bigvee_{\bar{\gamma}\in\{\top,\bot\}^s} \left(\bigwedge_{i<s} (\xi'_i \leftrightarrow \gamma_i) \wedge \bigwedge_{\tau\in\Upsilon} \tau\left\langle\beta\left(\chi_0^\tau, \ldots, \chi_{r-1}^\tau, \gamma_0, \ldots, \gamma_{s-1}, \zeta'_0, \ldots, \zeta'_{t-1}\right)\right\rangle \right)$$

The respective bounds are easily proved by an induction on the cases. □

[ix] The order types are mutually exclusive.

Corollary 1. $\mathrm{LTL}(\oplus_{\sim}, \oplus_{\not\sim}, \ominus_{\sim}, \ominus_{\not\sim})$ *is expressively complete w.r.t* $\mathrm{FO}^2(<, \lesssim)$.

The next interesting question about the logic is decidability, it turns out that the logic is decidable.

Proposition 2. $\mathrm{FO}^2(<, \lesssim)$ *is decidable in* NEXPTIME.

Proof. [Bojanczyk et al., 2006] shows that the satisfiability of $\mathrm{FO}^2(<, \sim)$ is decidable in NEXPTIME, where $\sim$ is an equivalence relation, We interpret $\mathrm{FO}^2(<, \lesssim)$ in $\mathrm{FO}^2(<, \sim)$ by the following translation,

$$\begin{array}{ll} \ulcorner a(x) \urcorner := a(x) & \ulcorner x = y \urcorner := x = y \\ \ulcorner x < y \urcorner := x < y & \ulcorner x \lesssim y \urcorner := x < y \wedge x \sim y \\ \ulcorner \neg\varphi \urcorner := \neg \ulcorner \varphi \urcorner & \ulcorner \varphi_1 \vee \varphi_2 \urcorner := \ulcorner \varphi_1 \urcorner \vee \ulcorner \varphi_2 \urcorner \\ \ulcorner \exists x.\varphi \urcorner := \exists x. \ulcorner \varphi \urcorner & \end{array}$$

This completes the proof. □

Corollary 2. *Satisfiability of* $\mathrm{LTL}(\oplus_{\sim}, \oplus_{\not\sim}, \ominus_{\sim}, \ominus_{\not\sim})$ *is in* NEXPTIME.

2 Discussion

Diamonds are not sufficient to specify properties over discrete linear time. We can enhance the expressiveness of our temporal logic by adding modalities for YESTERDAY and TOMORROW. We can redo the proofs and show that they translate to $\mathrm{FO}^2(<, \prec, \lesssim, \precsim)$ [x].But the satisfiability problem for $\mathrm{FO}^2(<, \prec, \lesssim, \precsim)$ is as hard as reachability in vector addition systems [Bojanczyk et al., 2006]. The situation is worse when we go for binary modalities like UNTIL, because of the following.

Proposition 3 ([Bojanczyk et al., 2006])**.** *Satisfiability of* $\mathrm{FO}^3(<, \lesssim)$ *is undecidable.*

We can redo the above proof and show that,

Proposition 4. *Satisfiability of* $\mathrm{FO}^3(<, \lesssim)$ *is undecidable.*

Another aspect is to refine the order $\lesssim$, if we see each local process as a collection of threads. The order $<$ stands for the global ordering of the events in each process, $\lesssim$ is the collection of orderings of each local process and another order $\lesssim'$ is the collection of threads in each local process and hence $\lesssim'$ has to be compatible with $\lesssim$ and again a union of chains. But, defining a decidable temporal logic is hard, since

[x] The trivial way to get expressive completeness with respect to $\mathrm{FO}^2(<, \prec, \lesssim, \precsim)$ is to add a modality for each order type definable, which in turn corresponds to taking the product of modalities definable in each oder (for instance, the order types definable by the vocabulary $(<, \prec)$ in two variables correspond to the modalities YESTERDAY, TOMORROW, PAST, FUTURE, DISTANT PAST, DISTANT FUTURE) interpreted in the obvious way.

Proposition 5 ([Björklund and Bojanczyk, 2007]). *Satisfiability of* $\mathrm{FO}^2(<, \lesssim, \lesssim')$ *, where* $\lesssim'$ *is compatible with* $\lesssim$ *and is a union of chains, is undecidable.*[xi]

It may be noted that in principle the temporal logic which is introduced here can be used for model-checking. The histories are of the form $(w, \lesssim)$ where $w \in \Sigma^*$ is a word and $\lesssim$ is as previously described. A collection of such histories can be recognized by the following automaton, we denote by $\precsim$ the Hasse covering relation of $\lesssim$. We say $i \in [|w|]$ is $\lesssim$-minimal if $\neg\exists j \in [|w|]$. $j \lesssim i$, similarly i is $\lesssim$-maximal if $\neg\exists j \in [|w|]$. $i \lesssim j$[xii].

Definition 1 ([Björklund and Schwentick, 2007]). *A Class memory automaton* A *is a six tuple* $A = (Q, \Sigma, \Delta, q_0, F_l, F_g)$ *where* Q *is a finite set of states,* q_0 *is the initial state,* $F_l \subseteq Q$ *is a set of local accepting states,* $F_g \subseteq F_l$ *is a set of global accepting states and* $\Delta \subseteq Q \times (Q \cup \{\bot\}) \times \Sigma \times Q$ *is the transition relation.*

A run ρ of the automaton A on a given word $\mathbf{w} = (w, \lesssim)$, where $w = a_1 a_2 \dots a_n$ is a sequence $q_0 q_1 \dots q_n$ such that q_0 is the initial state and for all $i \in [n]$ there is a transition $\delta_i = (p, p', a, q)$ such that (i) $p = q_{i-1}$ and $q = q_i$ (ii) $a = a_i$ (iii) if i is $\lesssim$-minimal then p' should be $\bot$. Else there is a $j \in [n]$ such that $j \precsim i$, and $p' = q_j$. The run ρ is said to be accepting if $\{q_i \mid i \text{ is } \lesssim\text{-maximal}\} \subseteq F_l$ and $q_n \in F_g$.

Example 1. The printer system can be modelled by the automaton in the following way. $A = (Q, \Sigma, \Delta, q_0, F_l, F_g)$ where $\Sigma = \{w, p, q, r\}$. The states of the automaton are $Q = \{s_0, s_r, s_q, s_p\}$. The accepting states $F_l = F_g = \{s_p\}$ and the initial state is s_0. The transition relation contains the following tuples $\Delta = \{(s, s', w, s') \mid s, s' \in Q\} \cup \{(s, s_0, r, s_r) \mid s \in Q\} \cup \{(s, s_r, q, s_q) \mid s \in Q\} \cup \{(s, s_q, p, s_p) \mid s \in Q\}$.

Class memory automaton is expressively equivalent to $\mathrm{EMSO}^2(<, \prec, \lesssim, \precsim)$ and its emptiness checking is decidable. Any LTL formula can be translated to a CMA in 2-Dexptime and can be model-checked. But the complexity of emptiness checking of CMA is as hard as Petri net reachability making it intractable.

3 Data Languages

Our logic and automaton work over structures of the form $(w, \lesssim)$, where $\lesssim$ is an extra-linguistic object, in the sense that it lies outside the level of abstraction of the alphabet. Even though in logic it is viable and convenient to consider such objects, in reality it is common that such informations are represented at the level of abstraction of the alphabet[xiii]. Examples are time-stamps in distributed systems, nonce-s in security protocols, process id-s in schedulers, ID attributes in XML documents etc.

[xi] Satisfiability of $\mathrm{FO}^2(<, \lesssim, \lesssim')$ is undecidable when $\lesssim'$ is not compatible with $\lesssim$ but still compatible with $<$. Since given φ we can construct the sentence $\varphi' \equiv \varphi \wedge \forall xy.\ x \lesssim' y \to x \lesssim y$.

[xii] $[n]$ for $n \in \mathbb{N}$ stands for the set $\{1, \dots n\}$.

[xiii] It may very well be the case that such informations are unavailable outside the realm of language.

It is clear that since $\lesssim$ has to represent unbounded-ly many processes, any alphabet which is going to code it up, has to be unbounded. It can be done by introducing a countably infinite alphabet with suitable relations and operations on the alphabet[xiv]. Hence the words, called data-words, are going to be over $\Sigma \times \Delta$, where Σ is finite and Δ is countably infinite. A collection of data-words is called a data language. For instance the semantics of our logic can be given in terms of words over $\Sigma \times \Delta$ where Δ has countably many arbitrarily long chains (for example $\Delta = \mathbb{N} \times \mathbb{N}$ with the relation $(x, y) \lesssim (x', y') \Leftrightarrow x = x' \wedge y < y'$).

We conclude by pointing to a comprehensive survey on Data languages [Segoufin, 2006].

References

[Alur et al., 2004] Alur, R., Etessami, K., and Madhusudan, P. (2004). A temporal logic of nested calls and returns. In Jensen, K. and Podelski, A., editors, *TACAS*, volume 2988 of *Lecture Notes in Computer Science*, pages 467–481. Springer.

[Alur and Madhusudan, 2006] Alur, R. and Madhusudan, P. (2006). Adding nesting structure to words. In Ibarra, O. H. and Dang, Z., editors, *Developments in Language Theory*, volume 4036 of *Lecture Notes in Computer Science*, pages 1–13. Springer.

[Björklund and Bojanczyk, 2007] Björklund, H. and Bojanczyk, M. (2007). Shuffle expressions and words with nested data. In Kucera, L. and Kucera, A., editors, *MFCS*, volume 4708 of *Lecture Notes in Computer Science*, pages 750–761. Springer.

[Björklund and Schwentick, 2007] Björklund, H. and Schwentick, T. (2007). On notions of regularity for data languages. In Csuhaj-Varjú, E. and Ésik, Z., editors, *FCT*, volume 4639 of *Lecture Notes in Computer Science*, pages 88–99. Springer.

[Bojanczyk et al., 2006] Bojanczyk, M., David, C., Muscholl, A., Schwentick, T., and Segoufin, L. (2006). Two-variable logic on words with data. In *LICS*, pages 7–16. IEEE Computer Society.

[Demri and Lazic, 2006] Demri, S. and Lazic, R. (2006). Ltl with the freeze quantifier and register automata. In *LICS '06: Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science*, pages 17–26, Washington, DC, USA. IEEE Computer Society.

[Etessami et al., 1997] Etessami, K., Vardi, M. Y., and Wilke, T. (1997). First-order logic with two variables and unary temporal logic. In *Logic in Computer Science*, pages 228–235.

[Kamp, 1968] Kamp, H. (1968). *On tense logic and the theory of order*. PhD thesis, UCLA.

[Pnueli, 1977] Pnueli, A. (1977). The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57, Providence, Rhode Island. IEEE.

[Segoufin, 2006] Segoufin, L. (2006). Automata and logics for words and trees over an infinite alphabet. In Ésik, Z., editor, *CSL*, volume 4207 of *Lecture Notes in Computer Science*, pages 41–57. Springer.

[Stockmeyer, 1974] Stockmeyer, L. (1974). The complexity of decision problems in automata theory and logic. PhD thesis, TR 133, M.I.T., Cambridge.

[xiv] Equality is the most essential, prefix relation, arithmetical predicates etc. are other examples, though these have not yet been studied.