Symbolic Synthesis for LTL$_f$+ Obligations

Giuseppe De Giacomo[1] , **Christian Hagemeier**[1] , **Daniel Hausmann**[2] , **Nir Piterman**[3]
[1]University of Oxford, UK
[2]University of Liverpool, UK
[3]University of Gothenburg and Chalmers University of Technology, Sweden
hausmann@liverpool.ac.uk, {christian.hagemeier,giuseppe.degiacomo}@cs.ox.ac.uk,
piterman@chalmers.se

Abstract

We study synthesis for obligation properties expressed in LTL$_f$+, the extension of LTL$_f$ to infinite traces. Obligation properties are positive Boolean combinations of safety and guarantee (co-safety) properties and form the second level of the temporal hierarchy of Manna and Pnueli. Although obligation properties are expressed over infinite traces, they retain most of the simplicity of LTL$_f$. In particular, we show that they admit a translation into symbolically represented deterministic weak automata (DWA) obtained directly from the symbolic deterministic finite automata (DFA) for the underlying LTL$_f$ properties on trace prefixes. We show that synthesis for LTL$_f$+ obligation properties is theoretically highly efficient – solvable in linear time once the DWA is constructed. We investigate several symbolic algorithms for solving DWA games that arise in the synthesis of obligation properties and evaluate their effectiveness experimentally. Overall, the results indicate that synthesis for LTL$_f$+ obligation properties can be performed with virtually the same effectiveness as LTL$_f$ synthesis.

1 Introduction

Linear Temporal Logic (LTL) (Pnueli 1977) and its finite-trace variants are commonly used in computer science and artificial intelligence, e.g., in planning for temporally extended goals and declarative control knowledge (Bacchus and Kabanza 1998; De Giacomo and Vardi 1999; Bacchus and Kabanza 2000; Calvanese, De Giacomo, and Vardi 2002; Baier, Fritz, and McIlraith 2007; Gerevini et al. 2009).

Reactive synthesis concerns the automatic construction of programs (typically called *strategies*) from temporal specifications for systems (e.g. agents, processes, protocols, controllers, or robots) that interact with their environments during execution (Pnueli and Rosner 1989; Finkbeiner 2016; Ehlers et al. 2017). It is closely related to strong planning for temporally extended goals in fully observable nondeterministic domains (Cimatti et al. 2003; Bacchus and Kabanza 1998; Calvanese, De Giacomo, and Vardi 2002; Baier, Fritz, and McIlraith 2007; Gerevini et al. 2009; De Giacomo and Rubin 2018). Significant advances in reactive synthesis have been achieved using the GR(1) fragment of LTL (Piterman, Pnueli, and Sa'ar 2006), and by restricting attention to finite traces using LTL$_f$ (De Giacomo and Vardi 2015). These successes were largely based on symbolic handling of temporal formulas, which substantially enhances scalability and computational capacity.

Recently, the symbolic techniques underlying LTL$_f$ have been extended to infinite traces through the logic LTL$_f$+ (Aminof et al. 2025b). The key promise of LTL$_f$+ is that finite automata (as used in LTL$_f$) and the symbolic representations that efficiently support them can also be leveraged in the setting of infinite traces. LTL$_f$+ builds on the so-called *Manna-Pnueli* (or *safety-progress*) *hierarchy* of temporal formulas, originally introduced in (Lichtenstein, Pnueli, and Zuck 1985) and subsequently developed in detail by Manna and Pnueli in (Manna and Pnueli 1990).

In this paper, we concentrate on synthesis for the lowest levels of this hierarchy, namely *safety* and *guarantee* (also known as *co-safety*) properties, as well as their Boolean closure, referred to as *obligation* properties, which are our main focus. Obligation properties are very common in practice. For instance, among the 55 specification patterns of Dwyer et al. (Dwyer, Avrunin, and Corbett 1998), 25 are obligation properties. Similarly, Somenzi & Bloem's compilation of 25 LTL formulas "found in the literature" (Somenzi and Bloem 2000) contains 13 obligation properties. Moreover, even in the LTL$_f$ setting, assumptions about the environments are often forms of obligations (Aminof et al. 2025a).

Our contributions are: (i) a symbolic translation of obligation LTL$_f$+ formulas into deterministic weak automata (DWA); (ii) a reduction of obligation LTL$_f$+ synthesis to DWA games, solvable in linear symbolic time once the DWA is built; (iii) a novel symbolic algorithm solving DWA games by alternating safety and reachability fixpoint computations; and (iv) an implementation and experimental evaluation showing that synthesis for obligation LTL$_f$+ matches the practical efficiency of LTL$_f$ synthesis.

An extended version is available (De Giacomo et al. 2026).

2 Obligations and Weak Automata

We recall basic notions of temporal logic and ω-automata.

LTL$_f$ and LTL$_f$+. Fix a set AP of atomic propositions. LTL formulas $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \psi$ (where $p \in AP$) are interpreted over infinite traces. In contrast, LTL$_f$ uses the same syntax interpreted over finite traces. Formulas of LTL$_f$+ are generated by the grammar

$$\Psi, \Psi' ::= \forall\Phi \mid \exists\Phi \mid \forall\exists\Phi \mid \exists\forall\Phi \mid \Psi \vee \Psi' \mid \Psi \wedge \Psi' \mid \neg\Psi$$

where Φ is an LTL$_f$ formula. We refer to formulas of the form $\mathbb{Q}\Phi$ as *finite-trace components*. We let $[\Phi] \subseteq (2^{AP})^*$ denote

the set of finite traces satisfying Φ. For $R \subseteq (2^{AP})^*$, let $\exists R$ ($\forall R$) denote the infinite traces with some (resp. every) prefix in R, and $\forall\exists R$ ($\exists\forall R$) those with infinitely many (resp. all but finitely many) prefixes in R. We evaluate LTL$_f$+ formulas Ψ over infinite traces using the extension $[\Psi] \subseteq (2^{AP})^\omega$ defined inductively by $[\Psi \vee \Psi'] = [\Psi] \cup [\Psi']$, $[\Psi \wedge \Psi'] = [\Psi] \cap [\Psi']$, $[\neg\Psi] = (2^{AP})^\omega \setminus [\Psi]$, and $[\mathbb{Q}\Phi] = \mathbb{Q}[\Phi]$ where $\mathbb{Q} \in \{\exists, \forall, \forall\exists, \exists\forall\}$.

LTL$_f$+ captures exactly the first-order definable infinite-trace properties (Aminof et al. 2025b). The *obligation fragment* restricts components to $\exists\Phi$ and $\forall\Phi$. It defines exactly the same obligation properties over infinite traces as LTL.

We note that our algorithms extend directly to the obligation fragment of PPLTL+ (Aminof et al. 2025b), the equally expressive logic in which finite-trace components are formulated in Pure-Past LTL (PPLTL) rather than LTL$_f$. Since PPLTL admits singly-exponential translation to DFA, the bounds in Corollary 1 and Theorem 1 become singly exponential for PPLTL+ obligation formulas.

Weak automata. Recall that a *Büchi automaton* $\mathcal{A} = (T, F)$ consists of a transition system $T = (\Sigma, Q, \iota, \delta)$ and a set $F \subseteq Q$ of accepting states; it accepts an infinite word iff some run on it visits F infinitely often. Such an automaton is *weak* if every SCC S in it is either entirely accepting ($S \subseteq F$) or entirely rejecting ($S \cap F = \emptyset$) (Löding 2001). For deterministic weak automata, Büchi and co-Büchi acceptance coincide; we call such automata *DWA*. These are known to be closed under union, intersection, and complement.

Minimization. For every DWA with n states, a unique minimal equivalent DWA can be computed in time $\mathcal{O}(n \log n)$ by a linear-time pass followed by Hopcroft minimization (Löding 2001).

3 From Obligations to DWA

LTL$_f$+ obligation formulas can be translated into DWA:

Recall that obligation formulas are Boolean combinations of components $\mathbb{Q}\Phi$, where Φ is a finite-trace formula and $\mathbb{Q} \in \{\exists, \forall\}$ a guarantee or safety trace quantifier. We first translate individual components into equivalent DWA.

Given an LTL$_f$ formula Φ over AP, let $\mathcal{A}_\Phi = (T_\Phi, F)$ with $T_\Phi = (2^{AP}, Q, \iota, \delta)$ denote an equivalent DFA of size $2^{2^{\mathcal{O}(|\varphi|)}}$, represented symbolically. Assume $\iota \notin F$ when $\mathbb{Q} = \exists$ and $\iota \in F$ when $\mathbb{Q} = \forall$.

Let $T_\Phi^+ = (2^{AP}, Q, \iota, \delta^+)$ and $T_\Phi^- = (2^{AP}, Q, \iota, \delta^-)$ be obtained from T_Φ by turning accepting (respectively, rejecting) states into absorbing sinks: define $\delta^+(q, a) = q$ if $q \in F$ and $\delta^+(q, a) = \delta(q, a)$ otherwise, and $\delta^-(q, a) = q$ if $q \notin F$ and $\delta^-(q, a) = \delta(q, a)$ otherwise.

Lemma 1. *Let Φ be an LTL$_f$ formula and $\mathbb{Q} \in \{\exists, \forall\}$. Then there exists a DWA $\mathcal{A}_{\mathbb{Q}\Phi}$ equivalent to $\mathbb{Q}\Phi$ such that $\mathcal{A}_{\mathbb{Q}\Phi} = (T_\Phi^+, F_\Phi)$ if $\mathbb{Q} = \exists$, and $\mathcal{A}_{\mathbb{Q}\Phi} = (T_\Phi^-, F_\Phi)$ if $\mathbb{Q} = \forall$.*

Proof. (Sketch) Let $\mathcal{A}_\Phi = (T_\Phi, F_\Phi)$ be a DFA for Φ. If $\mathbb{Q} = \exists$, an infinite trace satisfies $\exists\Phi$ iff some finite prefix is accepted by $\mathcal{A}_\Phi$; making accepting states absorbing in T_Φ^+ yields a DWA that accepts exactly those runs that eventually remain in F_Φ. If $\mathbb{Q} = \forall$, a trace satisfies $\forall\Phi$ iff every prefix is accepted; making rejecting states absorbing in T_Φ^- ensures that leaving F is permanent, so acceptance coincides with staying in F_Φ forever. In both constructions, SCCs do not mix accepting and rejecting states due to the absorbing sinks, hence the automata are weak. □

This leads to the following transformation from obligation LTL$_f$+ formulas to weak automata.

Corollary 1. *Every obligation LTL$_f$+ formula Ψ can be translated into an equivalent DWA of size $2^{2^{\mathcal{O}(|\Psi|)}}$.*

Proof. Let $T_1 \otimes T_2$ denote the standard synchronous product of deterministic transition systems, with state set $Q_1 \times Q_2$, initial state (ι_1, ι_2), and transition $\delta_\otimes((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$. Translate each component $\mathbb{Q}\Phi$ using Lemma 1. Combine the resulting automata according to the Boolean structure of Ψ via the closure properties of DWA:

$$T_{\Psi_i \bullet \Psi_j} = T_{\Phi_i} \otimes T_{\Phi_j} \quad (\bullet \in \{\wedge, \vee\})$$
$$F_{\Psi_i \wedge \Psi_j} = F_{\Psi_i} \times F_{\Psi_j}$$
$$F_{\Psi_i \vee \Psi_j} = F_{\Psi_i} \times Q_{\Phi_j} \cup Q_{\Phi_i} \times F_{\Psi_j}$$

□

4 Synthesis via DWA

We reduce reactive synthesis for obligation LTL$_f$+ to DWA games. Given an obligation Ψ in positive normal form over components $\mathbb{Q}_i\Phi_i$: (i) build each $\mathcal{A}_{\mathbb{Q}_i\Phi_i}$ via Lemma 1; (ii) compose these automata along the Boolean structure of Ψ (as in the proof of Corollary 1); minimization can be applied at any stage; (iii) solve the resulting DWA game on $\mathcal{A}_\Psi$.

Theorem 1. *Synthesis for obligation LTL$_f$+ is decidable symbolically via DWA games in* 2ExpTime.

For $|\Psi| = n$, the game has size $n' \leq 2^{2^n}$ (independent of the number of components and Boolean structure of Ψ), and by Lemma 3 below it can be solved in symbolic time $\mathcal{O}(n')$ – matching the worst-case complexity of LTL$_f$ synthesis.

Solution of DWA games. DWA games are special Büchi (and co-Büchi) games and admit the classical $\mathcal{O}(n^2)$ nested-fixpoint algorithms. We recall the controllable predecessor operator on a game arena $A = (V, E)$ partitioned into system/environment nodes V_s, V_e, defined, for $W \subseteq V$, by

$$\mathsf{C}(W) = \{v \in V_s \mid E(v) \cap W \neq \emptyset\} \cup \{v \in V_e \mid E(v) \subseteq W\}$$

and the standard fixpoints

$$\text{Reach}(W, T) = \mu X.\, T \cup (W \cap \mathsf{C}(X)),$$
$$\text{Safe}(W, T) = \nu X.\, T \cup (W \cap \mathsf{C}(X)),$$
$$\text{Büchi}(F) = \nu X.\mu Y.\, (F \cap \mathsf{C}(X)) \cup \mathsf{C}(Y).$$

We measure complexity in *symbolic operations* ($\cap, \cup, \neg$ on BDDs). Then C takes $\mathcal{O}(1)$, Reach and Safe take $\mathcal{O}(|W|)$, and Büchi (resp. coBüchi) takes $\mathcal{O}(|W|^2)$ operations.

Alternating safety and reachability. We propose Algorithm 1, which solves DWA games *without* nested fixpoints, by alternating safety and reachability over a growing sequence of sets W_i. Starting from $W_0 = \emptyset$, iteration i sets $W_{2i+1} = \text{Safe}(F, W_{2i})$ and $W_{2i+2} = \text{Reach}(V, W_{2i+1})$.

Algorithm 1: SOLVESAFEREACH(V,F)

$i = 0; W_0 = \emptyset; W_{-2} = V$
while $W_{2i} \neq W_{2(i-1)}$ **do**
  $W_{2i+1} = \texttt{Safe}(F, W_{2i})$
  $W_{2i+2} = \texttt{Reach}(V, W_{2i+1})$
  $i = i + 1$
return W;

Since $W_{2i} \subseteq W_{2i+1} \subseteq W_{2(i+1)}$, later fixpoints operate on larger targets and converge faster. The result SOLVESAFEREACH(V, F) is the set of nodes from which the system can eventually stay forever in an accepting SCC.

Lemma 2. *Algorithm 1 solves weak games with n nodes, k accepting nodes and l SCCs with $\mathcal{O}(n + kl) \subseteq \mathcal{O}(n^2)$ symbolic operations, and yields memoryless winning strategies.*

Linear solution. Parameterizing Algorithm 1 with an SCC decomposition and restricting fixpoint computations to individual SCCs yields a linear-time variant (Algorithm SOLVEWEAKSCC, similar to (Amram et al. 2021)): process bottom SCCs in a bottom-up DAG traversal, computing Safe(SCC, W) if the SCC is accepting and Reach(SCC, W) otherwise.

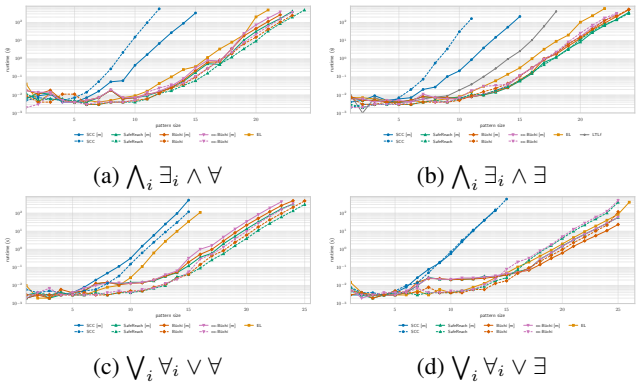
Lemma 3. SOLVEWEAKSCC *solves weak games with n nodes in symbolic time $\mathcal{O}(n)$ (Larsen et al. 2023), and yields memoryless winning strategies.*

5 Implementation and Experiments

We implemented all four algorithms (solving the DWA game using Büchi and co-Büchi solving, SOLVEWEAKSCC and SOLVESAFEREACH) by extending the LTL$_f$+ synthesis tool LydiaSyft+ (Hausmann et al. 2025). Each component $\mathbb{Q}\Phi$ is converted to a DWA via MONA's explicit DFA representation (Klarlund, Møller, and Schwartzbach 2002), and composition along the Boolean structure of Ψ uses two modes: (i) *component-wise* minimization followed by symbolic product, and (ii) *incremental* explicit products kept minimized while below threshold τ=256, switching to symbolic representation afterwards. The direct fixpoint algorithms (Büchi, co-Büchi, SafeReach) operate over the BDD representation. The SCC-based algorithm is theoretically more efficient but harder to realize here: existing linear-time SCC algorithms require a monolithic transition relation $T(x, x')$, whereas LydiaSyft+ uses a partitioned encoding that favors pre-image over post-image computation.

We evaluate our implementations on an M4 MacBook Air (12 GB RAM allocated, 10 min timeout), comparing against a more general synthesizer in LydiaSyft+ that reduces to Emerson-Lei games (Hausmann, Lehaut, and Piterman 2024; Hausmann et al. 2025), denoted "EL", and, where applicable, against LydiaSyft+'s LTL$_f$ synthesizer ("LTLf"). Variants with incremental minimization are marked with $[m]$.

A counter benchmark series adapted from (Hausmann et al. 2025) is dominated by DFA construction so all algorithms perform near-identically. In an additional experiment, we consider several formula patterns obtained by combining



(a) $\bigwedge_i \exists_i \wedge \forall$ (b) $\bigwedge_i \exists_i \wedge \exists$ (c) $\bigvee_i \forall_i \vee \forall$ (d) $\bigvee_i \forall_i \vee \exists$

Figure 1: Runtime of solvers on conjunction/disjunction patterns over $\varphi_i = \mathsf{F}((a_i \vee e_i) \wedge \mathsf{X}\,\mathsf{false})$. Solid lines = incremental minimization $[m]$; dashed = component-wise.

LTL$_f$ formulas of the form $\varphi_i = \mathsf{F}((e_i \vee a_i) \wedge \mathsf{X}\,\mathsf{false})$ with different Boolean operators and trace quantifiers, generalising a benchmark family from (Hausmann et al. 2025). The most striking finding is that the Büchi-based solvers scale to game arenas with roughly 2^{25} states, demonstrating that synthesis for obligation LTL$_f$+ can be performed at similar scales as state-of-the-art LTL$_f$ synthesis, despite operating over infinite traces.

We attribute the comparatively good performance of the EL-based solver for experiments on conjunctions of guarantuee properties (Figures 1(a) and 1(b)) to the fact that the EL solver constructs and solves generalized Büchi games in these experiments. Using this approach, the EL solver decomposes the overall objective into several sub-objectives that can be solved independently. An overall solution is obtained by composing the solutions for the sub-objectives. This apparently can lead to advantages over the Büchi-based algorithms which solve games with a single monolithic objective, which is simpler but may require more iterations until a fixpoint is obtained. The comparatively weak performance of the SCC-based algorithm is primarily due to implementation effects, in particular the automata representation used in LydiaSyft+, which is not well suited for efficient SCC computation. In instances where the SCC solver fails to produce a result within the allotted time, the construction of the monolithic transition relation typically fails because the BDD grows too large.

Notably, minimization has mixed effects on the SCC algorithm: in some cases (e.g., the $\bigvee \forall_i \vee \forall$ pattern) it improves performance, while in others (e.g., the $\bigwedge_i \exists_i \wedge \exists$ pattern) it degrades it. This likely reflects a trade-off between the reduced state space, which can improve efficiency, and the loss of the partitioned symbolic structure caused by minimization, which can make subsequent operations more expensive.

6 Conclusion

Synthesis for obligation LTL$_f$+ via DWA games matches the asymptotic complexity of LTL$_f$ synthesis, evidenced in practice by our experiments. We plan to extend the approach to recurrence and persistence properties, and to explore MTBDD-based representations (Duret-Lutz et al. 2025).

References

Aminof, B.; De Giacomo, G.; Di Stasio, A.; Francon, H.; Rubin, S.; and Zhu, S. 2025a. LTLf synthesis under environment specifications for reachability and safety properties. *Inf. Comput.* 303:105255.

Aminof, B.; De Giacomo, G.; Rubin, S.; and Vardi, M. Y. 2025b. LTLf+ and PPLTL+: Extending LTLf and PPLTL to infinite traces. In *IJCAI*.

Amram, G.; Bansal, S.; Fried, D.; Tabajara, L. M.; Vardi, M. Y.; and Weiss, G. 2021. Adapting behaviors via reactive synthesis. In *CAV*.

Bacchus, F., and Kabanza, F. 1998. Planning for temporally extended goals. *Ann. Math. Artif. Intell.* 22(1-2):5–27.

Bacchus, F., and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artif. Intell.* 116(1-2):123–191.

Baier, J. A.; Fritz, C.; and McIlraith, S. A. 2007. Exploiting procedural domain control knowledge in state-of-the-art planners. In *ICAPS*.

Calvanese, D.; De Giacomo, G.; and Vardi, M. Y. 2002. Reasoning about actions and planning in LTL action theories. In *KR*.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artif. Intell.* 1–2(147).

De Giacomo, G., and Rubin, S. 2018. Automata-theoretic foundations of fond planning for LTL_f/LDL_f goals. In *IJCAI*.

De Giacomo, G., and Vardi, M. Y. 1999. Automata-theoretic approach to planning for temporally extended goals. In *ECP*.

De Giacomo, G., and Vardi, M. Y. 2015. Synthesis for LTL and LDL on finite traces. In *IJCAI*.

De Giacomo, G.; Hagemeier, C.; Hausmann, D.; and Piterman, N. 2026. Symbolic synthesis for LTLf+ obligations. *CoRR* abs/2604.18532.

Duret-Lutz, A.; Zhu, S.; Piterman, N.; Giacomo, G. D.; and Vardi, M. Y. 2025. Engineering an LTL_f synthesis tool. In *CIAA*.

Dwyer, M. B.; Avrunin, G. S.; and Corbett, J. C. 1998. Property specification patterns for finite-state verification. In *FMSP*.

Ehlers, R.; Lafortune, S.; Tripakis, S.; and Vardi, M. Y. 2017. Supervisory control and reactive synthesis: a comparative introduction. *Discret. Event Dyn. Syst.* 27(2):209–260.

Finkbeiner, B. 2016. Synthesis of reactive systems. *Dependable Softw. Syst. Eng.* 45:72–98.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artif. Intell.* 173(5-6):619–668.

Hausmann, D.; Zhu, S.; Parretti, G.; Weinhuber, C.; Giacomo, G. D.; and Piterman, N. 2025. Emerson-Lei and Manna-Pnueli games for LTLf+ and PPLTL+ synthesis. In *KR*.

Hausmann, D.; Lehaut, M.; and Piterman, N. 2024. Symbolic solution of Emerson-Lei games for reactive synthesis. In *FoSSaCS*.

Klarlund, N.; Møller, A.; and Schwartzbach, M. I. 2002. MONA implementation secrets. *Int. J. Found. Comput. Sci.* 13(4):571–586.

Larsen, C. A.; Schmidt, S. M.; Steensgaard, J.; Jakobsen, A. B.; van de Pol, J.; and Pavlogiannis, A. 2023. A truly symbolic linear-time algorithm for SCC decomposition. In *TACAS*.

Lichtenstein, O.; Pnueli, A.; and Zuck, L. D. 1985. The glory of the past. In *Logic of Programs*, 196–218.

Löding, C. 2001. Efficient minimization of deterministic weak omega-automata. *Inf. Process. Lett.* 79(3):105–109.

Manna, Z., and Pnueli, A. 1990. A hierarchy of temporal properties. In *PODC*.

Piterman, N.; Pnueli, A.; and Sa'ar, Y. 2006. Synthesis of reactive(1) designs. In *VMCAI*.

Pnueli, A., and Rosner, R. 1989. On the synthesis of a reactive module. In *POPL*.

Pnueli, A. 1977. The temporal logic of programs. In *FOCS*.

Somenzi, F., and Bloem, R. 2000. Efficient Büchi automata for LTL formulae. In *CAV*.