LTL$_f$ Goal-oriented Service Composition - Supplementary Material

Primary Keywords: *(4) Theory; (8) Knowledge Representation/Engineering*

Proofs of Section "Solution Technique"

Proposition 1. *$a_0 \dots a_{m-1} \in \mathcal{L}(\mathcal{A}_\varphi)$ iff $(a_0, q_1) \dots (a_{m-1}, q_m) \in \mathcal{L}(\mathcal{A}_{\mathsf{act}})$, for some $q_1 \dots q_m$.*

Proof. By definition, $a_0 \dots a_{m-1} \in \mathcal{L}(\mathcal{A}_\varphi)$ iff there exist a run $r = q_0, q_1 \dots q_m$ s.t. for $1 \leq k \leq m$, $\delta(q_{k-1}, a_k) = q_k$ and $q_m \in F$. Consider the word $w' = (a_0, q_1) \dots (a_{m-1}, q_m)$. By construction of $\mathcal{A}_{\mathsf{act}}$, w' induces a run $r^d = r$. Since $q_m \in F$ by assumption, r^d is an accepting run for $\mathcal{A}_{\mathsf{act}}$, and therefore $w' \in \mathcal{L}(\mathcal{A}_{\mathsf{act}})$ is accepted. The other direction follows by construction because, if $(a_0, q_1) \dots (a_{m-1}, q_m) \in \mathcal{L}(\mathcal{A}_{\mathsf{act}})$, then by construction $q_1 \dots q_m$ is a run of $\mathcal{A}_\varphi$ over word $a_0 \dots a_{m-1}$, and since $q_m \in F$ by assumption $a_0 \dots a_{m-1} \in \mathcal{L}(\mathcal{A}_\varphi)$. ☐

Proposition 2. *Let h be a history with $\mathcal{C}$ and φ be a specification. Then, h is successful iff there exist a word $w \in A'^*$ such that $h = \tau_{\varphi,\mathcal{C}}(w)$ and $w \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{C}})$.*

Proof. We prove both directions of the equivalence separately.

($\Leftarrow$) Let w be a word $w = (a_0, q_1, o_0, \sigma_{o_0}) \dots (a_{m-1}, q_m, o_{m-1}, \sigma_{o_{m-1}}) \in A'^*$ Assume $w \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{C}})$. We have that the induced run $r = (q_0, \sigma_{10} \dots \sigma_{n0}), \dots, (q_m, \sigma_{1m} \dots \sigma_{nm})$ over $\mathcal{A}_{\varphi,\mathcal{C}}$ is accepting. This means that $q_m \in F$ and $\sigma_{im} \in F_i$ for all i. Now consider the history $h = \tau_{\varphi,\mathcal{C}}(w) = (\sigma_{10} \dots \sigma_{n0}), (a_0, o_0), \dots (\sigma_{1m} \dots \sigma_{nm})$. For every $0 \leq k < m$, we have by definition of δ' that $\delta_i(\sigma_{i,k}, a_k) = \sigma_{i,k+1}$ if $i = o_k$, and $\sigma_{i,k} = \sigma_{i,k+1}$ otherwise. Moreover, $o_k \in \{1 \dots n\}$, $a_k \in A$ and $\sigma_{ik} \in \Sigma_i$, for all $i \in \{1, \dots, n\}$ and $0 \leq k < m$. Therefore, h is a valid history. Moreover, h is a successful history because $q_m \in F$ iff $a_0, \dots, a_{m-1} \models \varphi$, and $\sigma_{im} \in F_i$ for all i.

($\Rightarrow$) Assume that $h = (\sigma_{10} \dots \sigma_{n0}), (a_0, o_0), \dots, (\sigma_{1m} \dots \sigma_{nm})$ is a successful history with $\mathcal{C}$ Then we both have that (*i*) $\mathsf{actions}(h) = a_0, \dots, a_{m-1} \models \varphi$ and (*ii*) $\sigma_{im} \in F_i$ for all i. By construction of the NFA $\mathcal{A}_\varphi$, proposition (*i*) implies that there exists a run $q_0, \dots, q_m$ where $q_m \in F$. Moreover, let $\sigma_1, \dots, \sigma_m$ be the sequence of service states s.t. $\sigma_k = \sigma_{i,k}$ (where $i = o_k$), for $1 \leq k \leq m$. Now consider the sequence $w = (a_0, q_1, o_0, \sigma_1), \dots, (a_{m-1}, q_m, o_{m-1}, \sigma_m)$. By construction, we have $\tau_{\varphi,\mathcal{C}}(w) = h$. Moreover, from proposition (*i*) and proposition (*ii*), it follows that $w \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{C}})$. This concludes the proof. ☐

DFA *games* are games between two players, here called respectively the *environment* and the *controller*, that are specified by a DFA. We have a set of $\mathcal{X}$ of *uncontrollable symbols*, which are under the control of the environment, and a set $\mathcal{Y}$ of *controllable symbols*, which are under the control of the controller. A *round* of the game consists of both the controller and the environment choosing the symbols they control. A (complete) *play* is a word in $\mathcal{X} \times \mathcal{Y}$ describing how the controller and environment set their propositions at each round till the game stops. The *specification* of the game is given by a DFA $\mathcal{G}$ whose alphabet is $\mathcal{X} \times \mathcal{Y}$. A play is *winning* for the controller if such a play leads from the initial to a final state. A *strategy* for the controller is a function $f : \mathcal{X}^* \to \mathcal{Y}$ that, given a history of choices from the environment, decides which symbols $\mathcal{Y}$ to pick next. In this context, a history has the form $w = (X_0, Y_0) \cdots (X_{m-1}, Y_{m-1})$. Let us denote by $w_{\mathcal{X}}|_k$ the sequence projected only on $\mathcal{X}$ and truncated at the k-th element (included), i.e., $w_{\mathcal{X}}|_k = X_0 \cdots X_k$. A *winning strategy* is a strategy $f : \mathcal{X}^* \to \mathcal{Y}$ such that for all sequences $w = (X_0, Y_0) \cdots (X_{m-1}, Y_{m-1})$ with $Y_i = f(w_{\mathcal{X}} \,|_k)$, we have that w leads to a final state of our DFA game. The *realizability* problem consists of checking whether there exists a *winning strategy*. The *synthesis* problem amounts to actually computing such a strategy.

We now give a sound and complete technique to solve realizability for DFA games. We start by defining the *controllable preimage* $PreC(\mathcal{E})$ of a set $\mathcal{E}$ of states $\mathcal{E}$ of $\mathcal{G}$ as the set of states s such that there exists a choice for symbols $\mathcal{Y}$ such that for all choices of symbols $\mathcal{X}$, game $\mathcal{G}$ progresses to states in $\mathcal{E}$. Formally, $PreC(\mathcal{E}) = \{s \in S \mid \exists Y \in \mathcal{Y}. \forall X \in \mathcal{X}. \delta(s, (X, Y)) \in \mathcal{E}\}$. Using such a notion, we define the set $Win(\mathcal{G})$ of winning states of a DFA game $\mathcal{G}$, i.e., the set formed by the states from which the controller can win the DFA game $\mathcal{G}$. Specifically, we define $Win(G)$ as a least-fixpoint, making use of approximates $Win_k(\mathcal{G})$ denoting all states where the controller wins in at most k steps: (1) $Win_0(\mathcal{G}) = F$ (the final states of $\mathcal{G}$); and (2) $Win_{k+1}(\mathcal{G}) = Win_k(\mathcal{G}) \cup PreC(Win_k(\mathcal{G}))$. Then, $Win(\mathcal{G}) = \bigcup_k Win_k(\mathcal{G})$. Notice that computing $Win(\mathcal{G})$ requires linear time in the number of states in $\mathcal{G}$.

Indeed, after at most a linear number of steps $Win_{k+1}(\mathcal{G}) = Win_k(\mathcal{G}) = Win(\mathcal{G})$. It can be shown that a DFA game $\mathcal{G}$ admits a winning strategy iff $s_0 \in Win(\mathcal{G})$ (De Giacomo and Vardi 2015).

The resulting strategy is a transducer $T = (\mathcal{X} \times \mathcal{Y}, Q', q'_0, \delta_T, \theta_T)$, defined as follows: $\mathcal{X} \times \mathcal{Y}$ is the input alphabet, Q' is the set of states, q'_0 is the initial state, $\delta_T : Q' \times \mathcal{X} \to Q'$ is the transition function such that $\delta_T(q, X) = \delta'(q, (X, \theta(q))$, and $\theta_T : Q \to \mathcal{Y}$ is the output function defined as $\theta_T(q) = Y$ such that if $q \in Win_{i+1}(\mathcal{G}) \setminus Win_i(\mathcal{G})$ then $\forall X.\delta(q, (X, Y)) \in Win_i(\mathcal{G})$ (De Giacomo and Vardi 2015).

Theorem 1. *The* realizability of service composition for LTL$_f$ goals *with community $\mathcal{C}$ for the satisfaction of an* LTL$_f$ *goal specification φ can be solved by checking whether $q'_0 \in Win(\mathcal{A}_{\varphi,\mathcal{C}})$.*

Proof. The proof is rather technical, therefore we first give an intuitive explanation. Soundness can be proved by induction on the maximum number of steps i for which the controller wins the DFA game from q'_0, building γ in a backward fashion such that it chooses $(a_k, o_k) \in A'$ that allows forcing the win in the DFA game (which exists by assumption). Completeness can be shown by contradiction, assuming that there exists an orchestrator γ that realizes φ with community $\mathcal{C}$, but that $q'_0 \notin Win(\mathcal{A}_{\varphi,\mathcal{C}})$; the latter implies that we can build an arbitrarily long history that is not successful, by definition of winning region, contradicting that γ realizes φ.

We now provide the full proof of the claim by separately proving the soundness and completeness of our approach.

Soundness. Assume $q'_0 = (q_0, \sigma_{10} \dots \sigma_{n0}) \in Win_m(\mathcal{A}_{\varphi,\mathcal{C}})$, i.e. the controller can win in at most m steps; we aim to show that there exists an orchestrator γ that realizes φ, i.e. all executions t are finite and successful.

We prove it by induction on the maximum number of steps i for which the controller wins the DFA game from q'_0, building γ in a backward fashion.

Base case ($k = 0$): assume $q'_0 \in F'$, i.e. q'_0 is already a goal state. Then, the problem is trivially realizable since the goal specification is already satisfied (the execution $t = (\sigma_{10} \dots \sigma_{n0})$ is a successful execution for any γ).

Inductive case: assume the claim holds for every state $q'_k \in Q'$ that can reach an accepting state in at most k steps, i.e. $q'_k = (q_k, \sigma_{1k} \dots \sigma_{nk}) \in Win_k(\mathcal{A}_{\varphi,\mathcal{C}})$, and let γ_k be the orchestrator that realizes the goal specification starting from such states. Let $\Delta Win_{k+1}(\mathcal{A}_{\varphi,\mathcal{C}}) = Win_{k+1}(\mathcal{A}_{\varphi,\mathcal{C}}) \setminus Win_k(\mathcal{A}_{\varphi,\mathcal{C}})$. Consider a state $q'_{k+1} = (q_{k+1}, \sigma_{1,k+1} \dots \sigma_{n,k+1}) \in \Delta Win_{k+1}(\mathcal{A}_{\varphi,\mathcal{C}})$. By construction, $q'_{k+1} \in PreC(Win_k(\mathcal{A}_{\varphi,\mathcal{C}}))$. This means that there exist a controllable symbol $Y \in \mathcal{Y}$ such that for all uncontrollable symbols $X \in \mathcal{X}$ we can reach a state in $Win_k(\mathcal{A}_{\varphi,\mathcal{C}})$, i.e. $\delta(q'_{k+1}, (X, Y)) = q'_k \in Win_k(\mathcal{A}_{\varphi,\mathcal{C}})$. Let γ_{k+1} be the orchestrator that behaves as γ_k in states in $Win_k(\mathcal{A}_{\varphi,\mathcal{C}})$, and $\gamma_{k+1}((\sigma'_{1,k+1} \dots \sigma'_{n,k+1})) = (a_{k+1}, o_{k+1})$, where $Y = (a_{k+1}, q'_{k+2}, o_{k+1})$, for every $\sigma'_{1,k+1} \dots \sigma'_{n,k+1}$ such that $(q'_{k+1}, \sigma'_{1,k+1} \dots \sigma'_{n,k+1}) \in \Delta Win_{k+1}(\mathcal{A}_{\varphi,\mathcal{C}})$. By the inductive hypothesis, we have that all finite executions t_k of γ_k, starting from some $(\sigma_{1k} \dots \sigma_{nk}) \in \Delta Win_k(\mathcal{A}_{\varphi,\mathcal{C}})$, are successful finite executions. To prove our claim, we only need to show that all t_{k+1}, starting from some state in $\Delta Win_{k+1}(\mathcal{A}_{\varphi,\mathcal{C}})$, are also successful executions. If $|t_{k+1}| \leq k$, e.g. when the adversary behaves cooperatively, then it holds by inductive hypothesis. For executions of length $k + 1$, this is the case because, by construction, we have $t_{k+1} = \sigma'_{z,k+1}, (a_{k+1}, o_{k+1}), t'$ for some execution t' of γ_k, some $(q'_{k+1}, \sigma'_{z,k+1}) \in \Delta Win_{k+1}(\mathcal{A}_{\varphi,\mathcal{C}})$, and $(a_{k+1}, o_{k+1}) = \gamma_{k+1}(\sigma'_{z,k+1})$. In other words, t_{k+1} is a valid finite execution of γ_{k+1}, and moreover, it is successful since t' is successful. Finally, we have that γ_m, by induction, is an orchestrator that can force the win of the game from q'_0.

Completeness. By contradiction, assume there exists an orchestrator γ that realizes φ with community $\mathcal{C}$, but that $q'_0 \notin Win(\mathcal{A}_{\varphi,\mathcal{C}})$. If $q'_0 \notin Win(\mathcal{A}_{\varphi,\mathcal{C}})$, then it means, by definition of $Win(\mathcal{A}_{\varphi,\mathcal{C}})$ and $PreC$, that for all $Y \in \mathcal{Y}$, there exist $X \in \mathcal{X}$ such that the successor state q'_1 is not in $Win(\mathcal{A}_{\varphi,\mathcal{C}})$. Therefore, we can recursively generate an arbitrarily long word $w = (X_0, Y_0) \dots (X_{m-1}, Y_{m-1})$, for any choice of $Y_0 \dots Y_{m-1}$, such that $w \notin \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{C}})$. Now consider the execution $t = (\sigma_{10} \dots \sigma_{n0}), (a_0, o_0), (\sigma_{11} \dots \sigma_{n1}) \dots (a_{m-1}, o_{m-1}), (\sigma_{1m} \dots \sigma_{nm})$, built as follows: for all $0 \leq k \leq m - 1$, $(a_k, o_k) = \gamma((\sigma_{10} \dots \sigma_{n0}) \dots (\sigma_{1k} \dots \sigma_{nk}))$, and for any $Y_k = (a_k, q_{k+1}, o_k)$, take $X_k = \sigma$ such that $w = (Y_0, X_0) \dots (Y_k, X_k) \notin \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{C}})$ and $\delta_{o_k}(\sigma_{o_k,k}, a_k) = \sigma$, and set $\sigma_{o_k,k} = \sigma$. In other words, we consider any valid execution t of γ where the successor state of the chosen service is taken according to the winning environment strategy (which is losing for the agent). By construction, t is an infinite execution of γ. Moreover, no prefix of t it is not successful, because w is not accepted by $\mathcal{A}_{\varphi,\mathcal{C}}$ and, by construction of $\mathcal{A}_{\varphi,\mathcal{C}}$, this means that either $\mathsf{actions}(h) \not\models \varphi$, or for some service $\mathcal{S}_i$, $\sigma_{ik} \notin F_i$. Since we assumed that γ realizes φ with community $\mathcal{C}$, but we can construct an execution t of γ that is not successful, we reached a contradiction. □

Theorem 2. *Let the composition problem be $\langle \varphi, \mathcal{C} \rangle$, and let the transducer T be a winning strategy over the game arena $\mathcal{A}_{\varphi,\mathcal{C}}$. Let γ_T be the orchestrator extracted from T, as defined above. Then, γ_T realizes φ with community $\mathcal{C}$.*

Proof. By definition, γ_T realizes φ with $\mathcal{C}$ if all its executions are finite successful traces. By contradiction, assume this is not the case, and that there exists an infinite execution $t = (\sigma_{10} \dots \sigma_{n0}), (a_0, o_0), \dots$, for which all finite prefixes t' of t are such that $\mathsf{actions}(t') \not\models \varphi$ and $last(\mathsf{states}(t')) \notin F_1 \times \dots \times F_n$. Since the set of states $\Sigma_1 \times \dots \times \Sigma_n$ is finite, it must be the case that a service configuration $\sigma_1, \dots \sigma_n$ is visited more than once in t. Consider the corresponding infinite trace produced by the strategy implemented by T while playing the game $\mathcal{A}_{\varphi,\mathcal{C}}$: $\tau = (q_0, \sigma_{10} \dots \sigma_{n0}), (a_0, q_1, o_0, \sigma_{o_0}), \dots$. By construction of $\mathcal{A}_{\varphi,\mathcal{C}}$, it follows that there exists an environment move X that forces the agent to loop infinitely and never reach the goal states in $\mathcal{A}_{\varphi,\mathcal{C}}$. But this contradicts the fact that T is a winning strategy that forces the game to reach an accepting state. □

Implementation and Applications

The LTL_f goal specification is a sequential goal, with the following actions: $cleaning, filmDeposition$, $resistCoating$, $exposure$, $development$, $etching$, $impurities_i mplantation$, $activation$, $resistStripping$, $assembly$, $testing$, and $packaging$.

The formula has the following form:

$$
\begin{aligned}
&\Diamond(cleaning \wedge \\
&\ \Diamond(filmDeposition \wedge \\
&\ \ \Diamond(resistCoating \wedge \\
&\ \ \ \Diamond(exposure \wedge \\
&\ \ \ \ \Diamond(development \wedge \\
&\ \ \ \ \ \Diamond(etching \wedge \\
&\ \ \ \ \ \ \Diamond(impurities_i mplantation \wedge \\
&\ \ \ \ \ \ \ \Diamond(activation \wedge \\
&\ \ \ \ \ \ \ \ \Diamond(resist_s tripping \wedge \\
&\ \ \ \ \ \ \ \ \ \Diamond(assembly \wedge \\
&\ \ \ \ \ \ \ \ \ \ \Diamond(testing \wedge \\
&\ \ \ \ \ \ \ \ \ \ \ \Diamond(packaging))))))))))))
\end{aligned}
$$

The sequence is truncated so to make the benchmarks more tractable.

Platform. The experiments have been run on an Ubuntu 22.04 machine, endowed with 12th Gen Intel(R) Core(TM) i7-1260P, with 16 CPU threads (12 cores) and 64GB of RAM. The JVM version is 14 for compatibility with MyND. The JVM maximum RAM was 16GB.

Running times

References

De Giacomo, G.; and Vardi, M. Y. 2015. Synthesis for LTL and LDL on Finite Traces. In *IJCAI*.

	Simple				OSA				PG				OSA+PG			
	TT	PT	EN	PS	TT	PT	EN	PS	TT	PT	EN	PS	TT	PT	EN	PS
Garden Bots System																
$h_{\max}$																
i0	14.601	20.121	147012	94	14.9	2.506	46209	306	15.147	3.393	—	—	15.425	4.161	51981	323
h_{ff}																
i0	14.055	31.143	145539	94	13.202	3.086	51376	306	14.38	5.518	—	—	15.127	4.436	42301	323
Electric Motor scenario																
$h_{\max}$																
e0	15.277	—	—	—	9.694	15.329	357667	195	10.305	—	—	—	12.829	—	—	—
e1	—	—	—	—	19.252	36.927	608938	351	—	—	—	—	—	—	—	—
e2	—	—	—	—	14.142	76.623	795607	351	—	—	—	—	—	—	—	—
e3	—	—	—	—	13.375	241.31	1570372	936	—	—	—	—	—	—	—	—
eu	17.785	—	N/A	N/A	17.689	—	N/A	N/A	21.745	—	N/A	N/A	11.17	112.97	N/A	N/A
h_{ff}																
e0	17.573	—	—	—	16.157	19.601	372855	195	10.074	—	—	—	14.324	—	—	—
e1	—	—	—	—	17.556	45.353	627812	273	—	—	—	—	—	—	—	—
e2	—	—	—	—	14.974	145.691	1006368	390	—	—	—	—	—	—	—	—
e3	—	—	—	—	13.476	—	—	—	—	—	—	—	—	—	—	—
eu	18.368	—	N/A	N/A	17.084	76.341	N/A	N/A	17.586	—	N/A	N/A	11.923	94.928	N/A	N/A
Chip Production scenario																
$h_{\max}$																
c1	5.805	0.042	171	19	6.268	0.02	59	34	5.083	0.023	62	15	7.037	0.024	65	31
c2	9.255	0.791	9892	38	10.736	0.18	1491	84	8.651	0.84	7502	33	9.799	0.378	3358	79
c3	17.256	91.894	297054	63	10.484	0.969	17765	160	17.619	111.212	254985	57	16.24	4.773	55363	153
c4	25.337	—	—	—	8.114	8.754	148589	270	19.841	—	—	—	24.548	183.926	654100	266
c5	—	—	—	—	35.207	92.093	912222	420	—	—	—	—	17.554	—	—	—
c6	—	—	—	—	36.318	—	—	—	—	—	—	—	—	—	—	—
h_{ff}																
c1	5.959	0.045	138	19	6.397	0.027	99	34	5.058	0.015	15	15	8.117	0.023	51	31
c2	8.162	0.722	8869	38	11.002	0.234	2158	84	7.994	0.321	1921	33	10.412	0.329	1580	79
c3	16.399	79.585	278034	63	12.633	1.55	22905	160	18.133	119.767	168536	57	16.725	4.651	38932	153
c4	24.074	—	—	—	7.961	10.951	161088	270	22.199	—	—	—	24.238	182.78	520057	266
c5	—	—	—	—	33.462	106.126	966889	420	—	—	—	—	17.999	—	—	—
c6	—	—	—	—	36.412	—	—	—	—	—	—	—	—	—	—	—

Table 1: Evaluation metrics over the considered scenario, using MyND with $h_{\max}$ and h_{ff} heuristics.