Skill Transfer for Temporal Task Specification: Supplementary Materials

Jason Xinyu Liu*[1], Ankit Shah*[1], Eric Rosen[1], Mingxi Jia[1], George Konidaris[1] and Stefanie Tellex[1]

I. The Implementation Details of LPOPL

LPOPL [1]'s DQN policy consists of a feedforward network with two hidden layers, each of which has 64 ReLU units. We used the same hyperparameters suggested in [1] for training, i.e., the learning rate is 0.0001; the size of the replay buffer is 25,000; 32 transitions are randomly sampled from the replay buffer for every update; the discount factor is 0.9; exploration decreases linearly from 1 to 0.02.

II. Example LTL Formulas

We provide LTL task specifications and their interpretations from the *Hard*, *Soft*, *Strictly Soft*, *No Orders*, and *Mixed* formula types.

Hard: Example formulas and their interpretations from the *Hard* type are as follows:

1) $\mathbf{F}workbench \wedge \mathbf{F}factory \wedge \mathbf{F}iron \wedge \mathbf{F}shelter \wedge \neg factory\ \mathbf{U}\ axe$: Visit $workbench$, $factory$, $iron$, $shelter$, and axe. Ensure that $factory$ is not visited before axe.
2) $\mathbf{F}toolshed \wedge \mathbf{F}bridge \wedge \mathbf{F}factory \wedge \mathbf{F}axe \wedge \neg bridge\ \mathbf{U}\ wood$: Visit $toolshed$, $bridge$, $factory$, axe, and $wood$. Ensure that $bridge$ is not visited before $wood$.
3) $\mathbf{F}wood \wedge \mathbf{F}axe \wedge \neg wood\ \mathbf{U}\ grass \wedge \neg grass\ \mathbf{U}\ workbench \wedge \neg workbench\ \mathbf{U}\ bridge$: Visit $bridge$, $workbench$, $grass$, $wood$, and axe. Ensure visiting $bridge$, $workbench$, $grass$, and $wood$ in that particular order. Objects that occur later in the sequence cannot be visited before any prior objects.

Soft: Example formulas and their interpretations from the *Soft* type are as follows:

1) $\mathbf{F}(bridge \wedge \mathbf{F}(factory \wedge \mathbf{F}(iron \wedge \mathbf{F}shelter)))$: Visit $bridge$, $factory$, $iron$, and $shelter$ in that sequence. The objects that occur later in the sequence may be visited before the prior objects, provided that they are visited at least once after the prior object has been visited.
2) $\mathbf{F}workbench \wedge \mathbf{F}(factory \wedge \mathbf{F}grass)$: Visit the $workbench$, $factory$, and $grass$. Visit $grass$ at least once after visiting the $factory$.
3) $\mathbf{F}(axe \wedge \mathbf{F}factory) \wedge \mathbf{F}workbench$: Visit axe, $factory$, and $workbench$. Ensure that $factory$ is visited at least once after axe.

Strictly Soft: Example formulas and their interpretations from the *Strictly Soft* type are identical to the *Soft* specifications, except they do not allow simultaneous satisfaction of multiple subtasks. The subtasks in the sequence must occur strictly after the prior subtask. This is enforced using nested operators next and finally $\mathbf{XF}a$ instead of $\mathbf{F}a$.

No Orders: These specifications only contain a list of subtasks to be completed. No temporal orders are enforced between any two subtasks.

1) $\mathbf{F}wood \wedge \mathbf{F}grass \wedge \mathbf{F}stone$: Collect $wood$, $grass$, $stone$ in no particular order.

Mixed: Example formulas and their interpretations from the *Mixed* type are as follows:

1) $\mathbf{F}toolshed \wedge \mathbf{F}factory \wedge \neg toolshed\ \mathbf{U}\ shelter \wedge \mathbf{F}(grass \wedge \mathbf{F}bridge)$: Visit the $toolshed$, $factory$, $shelter$, $grass$, and $bridge$. Ensure that $toolshed$ is not visited before the $shelter$ and $bridge$ is visited at least once after $grass$.
2) $\mathbf{F}grass \wedge \neg grass\ \mathbf{U}\ toolshed \wedge \mathbf{F}(factory \wedge \mathbf{XF}workbench)$: Visit $grass$, $toolshed$, $factory$, and $workbench$. Ensure that $grass$ is not visited before $toolshed$ and $workbench$ is visited at least once strictly after $factory$.
3) $\mathbf{F}iron \wedge \neg iron\ \mathbf{U}\ toolshed \wedge \mathbf{F}(shelter \wedge \mathbf{XF}wood)$: Visit $iron$, $toolshed$, $shelter$, and $wood$. Ensure that $iron$ is not visited before $toolshed$ and $wood$ is visited at least once strictly after $shelter$.

III. Additional Experimental Results

Learning Curves for Various Training Sets: We present learning curves of success rates for transferring policies learned on different specification types.

The learning curves for training on LTL tasks from the *Hard* training set with both *Relaxed* and *Constrained* edge-matching conditions are depicted in Figure 1.

The learning curves for training on LTL tasks from the *Soft* training set with both *Relaxed* and *Constrained* edge-matching conditions are depicted in Figure 2.

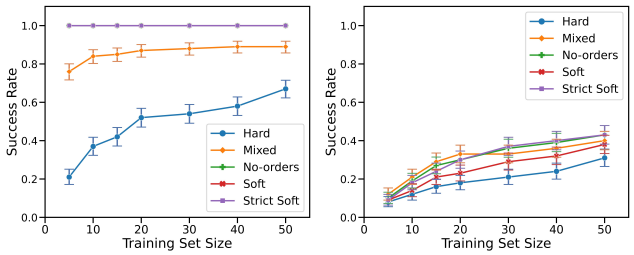
The learning curves for training on LTL tasks from the *Strictly Soft* training set with both *Relaxed* and *Constrained* edge-matching conditions are depicted in Figure 3.

The learning curves for training on LTL tasks from the *No Orders* training set are expected to share nearly identical trends as the learning curves from the other training sets.

Note that for training on each specification type, the learning curve trends are nearly identical to the learning curves of training on the *Mixed* specification types, as depicted in Figure 3 in the main paper. *Hard* specification types remain the most challenging specification types to transfer to.

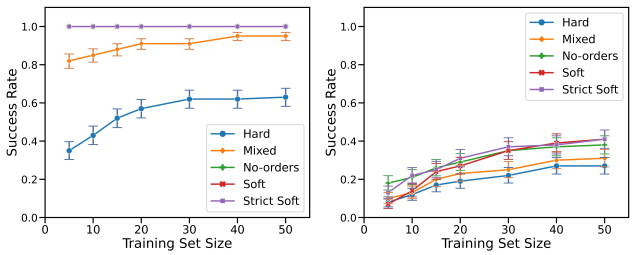
Transferability of Specification Types: We evaluated whether policies learned on different specification types are

*Equal contribution. [1] Brown University.



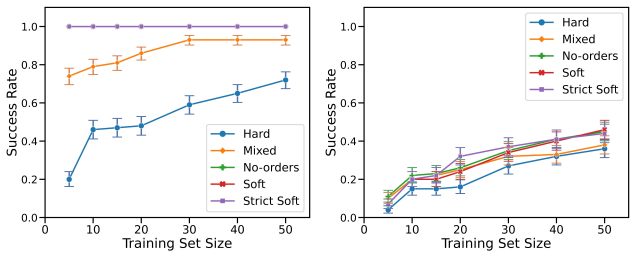
(a) *Relaxed* Edge Match (b) *Constrained* Edge Match

Fig. 1: Figure 1a depicts the success rates of transferring to five different specifications types using the *Relaxed* edge-matching condition after LTL-Transfer being trained on *Hard* training sets of various sizes. Figure 1b depicts the success rates with the *Constrained* edge-matching condition. Note that the error bars depict the 95% credible interval if the successful transfer was modeled as a Bernoulli distribution.



(a) *Relaxed* Edge Match (b) *Constrained* Edge Match

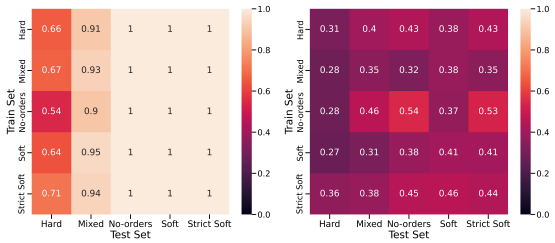
Fig. 2: Figure 2a depicts the success rates of transferring to five different specifications types using the *Relaxed* edge-matching condition after LTL-Transfer being trained on *Soft* training sets of various sizes. Figure 2b depicts the success rates with the *Constrained* edge-matching condition. Note that the error bars depict the 95% credible interval if the successful transfer was modeled as a Bernoulli distribution.



(a) *Relaxed* Edge Match (b) *Constrained* Edge Match

Fig. 3: Figure 3a depicts the success rates of transferring to five different specifications types using the *Relaxed* edge-matching condition after LTL-Transfer being trained on *Strictly Soft* training sets of various sizes. Figure 3b depicts the success rates with the *Constrained* edge-matching condition. Note that the error bars depict the 95% credible interval if the successful transfer was modeled as a Bernoulli distribution.

more capable of transferring to all other specification types. Figure 4a depicts the heatmap of success rates obtained by training on 50 specifications of the type indicated by the row and transferring to 100 specifications of the type indicated by the column while using the *Relaxed* edge-matching condition. Similarly, Figure 4b depicts the success rates using the *Constrained* edge-matching condition. No single specification type proved to be the best training set, thus providing evidence against the hypothesis that training with formulas conforming to certain formula templates leads to a greater success rate when transferring to all specification types. Further, training on all specification types leads to perfect transfer performance on *Soft*, *Strictly-Soft*, and *No-Orders* test set, thus providing further evidence for **H3** in the main paper.



(a) *Relaxed* Edge Match (b) *Constrained* Edge Match

Fig. 4: Heatmaps of success rates of LTL-Transfer using the *Relaxed* and *Constrained* edge-matching conditions, respectively, on various training and test specification types.

Failure Analysis: As described in the main paper, we logged the reason for the failure of each unsuccessful transfer attempt. There are three possible causes:

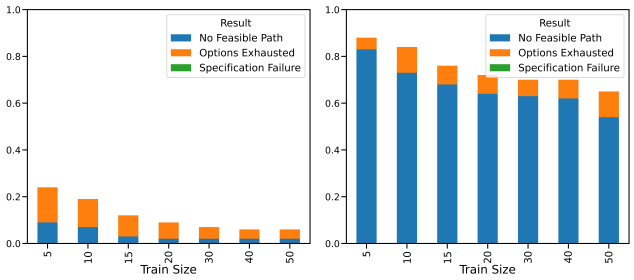
1) *Specification failure*: A constraint is violated during execution, and the reward machine progresses to an unrecoverable state.
2) *No feasible path*: After pruning the reward machine graph by removing infeasible edges, there are no paths connecting the start state to an accepting state with matching transition-centric options.
3) *Options exhausted*: During transferring, there are no further transition-centric options available to further progress the state of the reward machine.

Figure 5 depicts the relative frequency of the failure modes when LTL-Transfer is trained and tested on *mixed* task specifications. Note that with the *Relaxed* edge-matching condition not progressing the task after utilizing all available safe options is the primary reason for failure (Figure 5a) whereas, with the *Constrained* edge-matching condition, the absence of feasible paths connecting the start and the accepting state is the primary reason for failure (Figure 5b). Neither has specification failures due to violating a safety constraint.

IV. Selected Solution Trajectories in Simulation

Consider the case with *Mixed* training set with 5 formulas on *Map 0*. The training formulas are:

- $\mathbf{F}grass \wedge \mathbf{F}shelter \wedge \mathbf{F}(wood \wedge \mathbf{XF}workbench)$
- $\mathbf{F}toolshed \wedge \mathbf{F}workbench \wedge \mathbf{F}shelter \wedge (\neg toolshed\ \mathbf{U}\ shelter) \wedge \mathbf{F}(grass \wedge \mathbf{F}bridge)$



(a) *Relaxed* Edge Match (b) *Constrained* Edge Match

Fig. 5: Reasons for failed task execution after being trained and evaluated on the *Mixed* task specification datasets. Note that all values are depicted in fractions.

- $\mathbf{F}toolshed \wedge \mathbf{F}(shelter \wedge \mathbf{F}(axe \wedge \mathbf{F}wood))$
- $\mathbf{F}iron \wedge \mathbf{F}(shelter \wedge \mathbf{XF}(bridge \wedge \mathbf{XF}factory))$
- $\mathbf{F}factory$

One of the *Mixed* test formulas is $\varphi_{test} = \mathbf{F}workbench \wedge \mathbf{F}grass \wedge \mathbf{F}axe$. The reward machine for this task specification is depicted in Figure 6a. Given the training set of formulas and the use of the *Constrained* edge-matching condition, the start reward machine state is disconnected from all downstream states as no transition-centric options match the edge transitions, shown in Figure 6b. Therefore, LTL-Transfer does not attempt to solve the task and returns failure with the reason being *no feasible path*, i.e., a disconnected reward machine graph after removing infeasible edges.

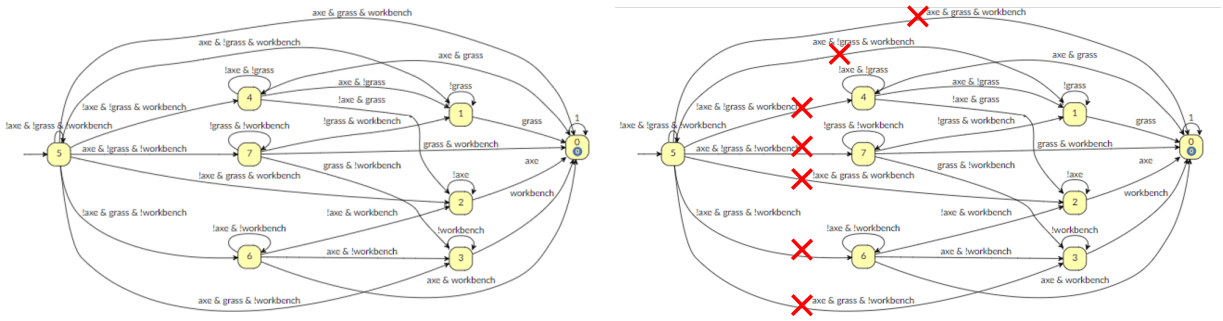
If the *Relaxed* edge-matching condition is used, there are matching transition-centric options for each edge, shown in Figure 6a. The trajectory produced by LTL-Transfer when transferring the policies is depicted in Figure 7. The robot collects all three requisite resources before it terminates the task execution. Further, note that the robot passes through a grid containing $wood$ as the specification does not explicitly prohibit it.

V. Robot Demonstration

The 50 test tasks executed on the robot are shown in Table I. The Proposition a represents a brown desk, b represents a white desk, c represents a couch, d represents a door, s represents a bookshelf, and k represents a kitchen counter.

References

[1] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Teaching multiple tasks to an RL agent using LTL," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018, pp. 452–461.



(a) (b)

Fig. 6: Figure 6a depicts the reward machine graph for the task specification $\varphi_{test} = \mathbf{F}workbench \wedge \mathbf{F}grass \wedge \mathbf{F}axe$, as well as all feasible edges matched by the *Relaxed* edge-matching condition. Note that all the edges have at least one matching transition-centric option. Figure 6b depicts the edges that do not have a compatible transition-centric option for the *Constrained* edge-matching condition.

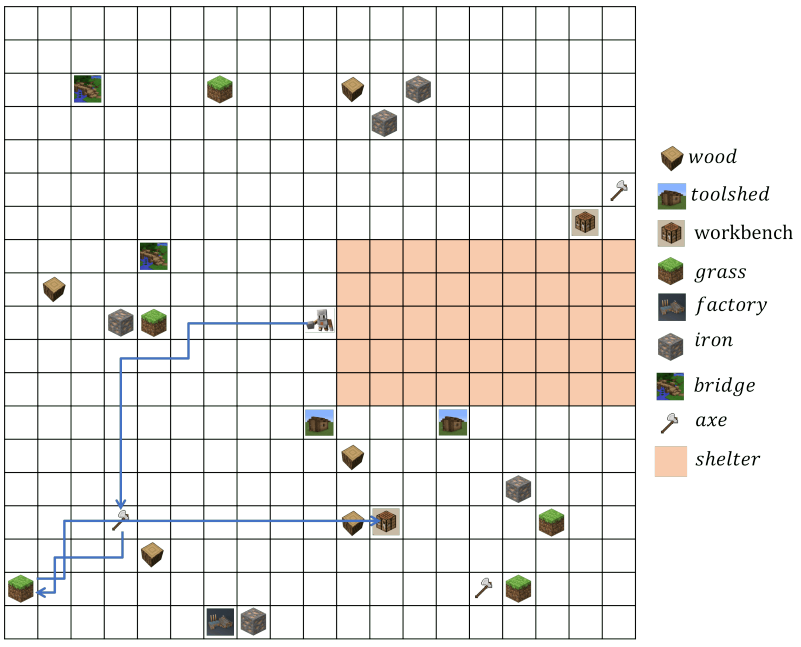


Fig. 7: Trajectory executed by the robot using LTL-Transfer to achieve the novel task specification $\varphi_{test} = \mathbf{F}workbench \wedge \mathbf{F}grass \wedge \mathbf{F}axe$.

TABLE I: Test Tasks Executed on the Robot

LTL Task Specification	Type of Task	Results
0. $\mathbf{F}a$	navigation	success
1. $\mathbf{F}a \wedge \mathbf{F}b$	navigation	success
2. $\mathbf{F}a \wedge \mathbf{F}b \wedge \mathbf{F}c$	navigation	success
3. $\mathbf{F}a \wedge \mathbf{F}b \wedge \mathbf{F}s$	navigation	success
4. $\mathbf{F}a \wedge \mathbf{F}b \wedge \mathbf{F}k$	navigation	success
5. $\mathbf{F}a \wedge \mathbf{F}b \wedge \mathbf{F}c \wedge \mathbf{F}d$	navigation	success
6. $\mathbf{F}a \wedge \mathbf{F}b \wedge \mathbf{F}c \wedge \mathbf{F}s$	navigation	success
7. $\mathbf{F}a \wedge \mathbf{F}b \wedge \mathbf{F}c \wedge \mathbf{F}k$	navigation	success
8. $\mathbf{F}a \wedge \mathbf{F}b \wedge \mathbf{F}c \wedge \mathbf{F}k \wedge \mathbf{F}s$	navigation	success
9. $\mathbf{F}(b \wedge \mathbf{F}(a \wedge \mathbf{F}(c \wedge \mathbf{F}d))))$	navigation	success
10. $\mathbf{F}(s \wedge \mathbf{F}a)$	fetch and deliver	success
11. $\mathbf{F}(s \wedge \mathbf{F}b)$	fetch and deliver	success
12. $\mathbf{F}(a \wedge \mathbf{F}b)$	navigation	success
13. $\mathbf{F}(b \wedge \mathbf{F}a)$	navigation	success
14. $\mathbf{F}(a \wedge \mathbf{F}(s \wedge \mathbf{F}c))$	fetch and deliver	success
15. $\mathbf{F}(b \wedge \mathbf{F}(s \wedge \mathbf{F}c))$	fetch and deliver	success
16. $\mathbf{F}(s \wedge \mathbf{F}(a \wedge \mathbf{F}c))$	fetch and deliver	success
17. $\mathbf{F}(a \wedge \mathbf{F}(b \wedge \mathbf{F}c))$	navigation	success
18. $\mathbf{F}(s \wedge \mathbf{F}(a \wedge \mathbf{F}(k \wedge \mathbf{F}a))))$	fetch and deliver	success
19. $\mathbf{F}(a \wedge \mathbf{F}(b \wedge \mathbf{F}(c \wedge \mathbf{F}d))))$	navigation	success
20. $\mathbf{F}(s \wedge \mathbf{XF}a)$	fetch and deliver	success
21. $\mathbf{F}(b \wedge \mathbf{XF}s)$	navigation	success
22. $\mathbf{F}(a \wedge \mathbf{XF}b)$	navigation	success
23. $\mathbf{F}(b \wedge \mathbf{XF}a)$	navigation	success
24. $\mathbf{F}(a \wedge \mathbf{XF}(b \wedge \mathbf{XF}c))$	navigation	success
25. $\mathbf{F}(a \wedge \mathbf{XF}(s \wedge \mathbf{XF}b))$	fetch and deliver	success
26. $\mathbf{F}(b \wedge \mathbf{XF}(s \wedge \mathbf{XF}a))$	fetch and deliver	success
27. $\mathbf{F}(s \wedge \mathbf{XF}(b \wedge \mathbf{XF}a))$	fetch and deliver	success
28. $\mathbf{F}(k \wedge \mathbf{XF}b)$	fetch and deliver	success
29. $\mathbf{F}(k \wedge \mathbf{XF}a)$	fetch and deliver	success
30. $\neg a \mathbf{U} s \wedge \mathbf{F}a$	fetch and deliver	success
31. $\neg b \mathbf{U} a \wedge \mathbf{F}b$	navigation	success
32. $\neg a \mathbf{U} b \wedge \mathbf{F}a$	navigation	success
33. $b \mathbf{U} a \wedge \neg c \mathbf{U} b \wedge \mathbf{F}c$	navigation	success
34. $b \mathbf{U} k \wedge \mathbf{F}b$	fetch and deliver	success
35. $\neg b \mathbf{U} c \wedge \mathbf{F}b$	navigation	success
36. $\neg a \mathbf{U} s \wedge \neg b \mathbf{U} a \wedge \mathbf{F}b$	fetch and deliver	success
37. $\neg s \mathbf{U} a \wedge \neg b \mathbf{U} s \wedge \mathbf{F}b$	fetch and deliver	success
38. $\neg b \mathbf{U} a \wedge \neg s \mathbf{U} b \wedge \mathbf{F}s$	navigation	success
39. $\neg a \mathbf{U} b \wedge \neg s \mathbf{U} a \wedge \mathbf{F}s$	navigation	success
40. $\mathbf{F}a \wedge \mathbf{F}(b \wedge \mathbf{F}c)$	navigation	success
41. $\mathbf{F}a \wedge \neg c \mathbf{U} b \wedge \mathbf{F}c$	navigation	success
42. $\mathbf{F}(a \wedge \mathbf{F}b) \wedge \neg c \mathbf{U} a \wedge \mathbf{F}c$	navigation	success
43. $\mathbf{F}a \wedge \mathbf{F}(b \wedge \mathbf{XF}c)$	navigation	success
44. $\mathbf{F}(a \wedge \mathbf{F}b) \wedge \neg c \mathbf{U} b \wedge \mathbf{F}c$	navigation	success
45. $\mathbf{F}(b \wedge \mathbf{F}a) \wedge \neg c \mathbf{U} b \wedge \mathbf{F}c$	navigation	success
46. $\mathbf{F}c \wedge (\neg s \mathbf{U} a \wedge \mathbf{F}s)$	navigation	success
47. $\mathbf{F}c \wedge (\neg a \mathbf{U} s \wedge \mathbf{F}a)$	navigation	success
48. $\mathbf{F}c \wedge (\neg s \mathbf{U} b \wedge \mathbf{F}s)$	navigation	success
49. $\mathbf{F}c \wedge (\neg b \mathbf{U} s \wedge \mathbf{F}b)$	navigation	success