Skill Machines: Temporal Logic Skill Composition in Reinforcement Learning

Geraud Nangue Tasse*, Devon Jarvis, Steven James and Benjamin Rosman

University of the Witwatersrand, Johannesburg, South Africa

ICLR WITS RAIL LAB

Solving temporal logic tasks specified by regular languages without further learning by composing skill primitives

PAPER

What we want

We want agents (e.g robots) that can **solve multiple complex tasks** in an environment (e.g real world) **without further learning**

Defining tasks is hard!
Reward design requires expert knowledge

Learning is slow!
Sample efficiency in RL is terrible

Curses of dimensionality

In general (for P environment propositions/sub-goals):

- *Spatial curse*: ***Eventually*** 2^{2^P}
- *Temporal curse*: ***Always*** 2^{2^P}

Task: "Navigate to a button ***and then*** to a cylinder ***while never*** entering blue regions"

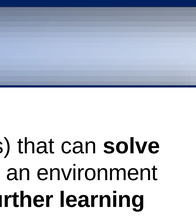
LTL: $(F(\bullet \wedge X(F\,\bullet))) \wedge (G\,\neg\bullet)$

Agent needs to:

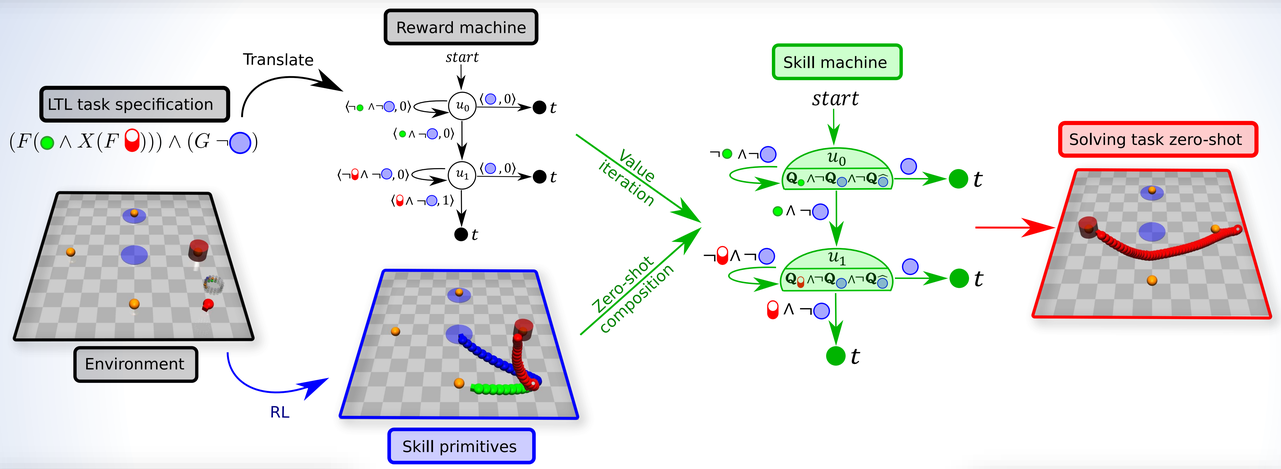
1. ***Eventually*** $\bullet \wedge \neg\bullet$
2. *Then* ***eventually*** $\bullet \wedge \neg\bullet$
3. *while* ***always*** $\neg\bullet$

Learned skills (Options)

- *"Go to a button"* $F\ \bullet$
- *"Go to a blue region"* $F\ \bullet$
- *"Go to the cylinder"* $F\ \bullet$



[1] R. T. Icarte et al., "Using reward machines for high-level task specification and decomposition in reinforcement learning," ICML, 2018
[2] G. Nangue Tasse et al., "World value functions: Knowledge representation for multitask reinforcement learning," RLDM, 2022.
[3] G. Nangue Tasse et al., "A Boolean task algebra for reinforcement learning" NeurIPS, 2020.



Solution: primitives

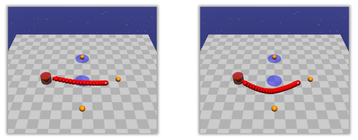
For spatial curse

- *Task primitives (M_p): Composable MDPs for **eventually** satisfying each proposition*
- *Skill primitives ($\boldsymbol{Q}_p$): WVFs for each task primitive*

For temporal curse

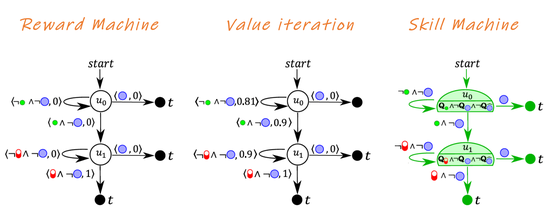
- *Constraints (C): Propositions to **always** keep True or False in an episode*
- *Augments the state space of task primitives*

E.g. $M_{\bullet}$ *where* $C = \{\bullet\}$



Reward machine to skill machine

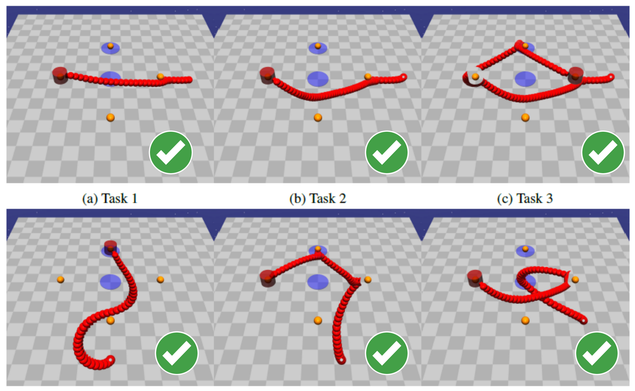
Plan over RM (e.g value iteration) then pick skills greedily



Zero-shot transfer

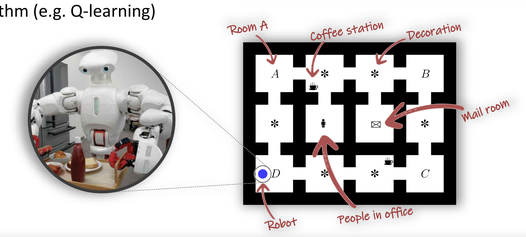
Task	Description — LTL
1	Navigate to a button and then to a cylinder. — $(F(\bullet \wedge X(F\,\bullet)))$
2	Navigate to a button and then to a cylinder while never entering blue regions — $(F(\bullet \wedge X(F\,\bullet))) \wedge (G\,\neg\bullet)$
3	Navigate to a button, then to a cylinder without entering blue regions, then to a button inside a blue region, and finally to a cylinder again. — $F(\bullet \wedge X(F((\bullet \wedge \neg\bullet) \wedge X(F((\bullet \wedge \bullet) \wedge X(F\bullet))))))$
4	Navigate to a button and then to a cylinder in a blue region. — $(F(\bullet \wedge X(F\,\bullet \wedge \bullet)))$
5	Navigate to a cylinder, then to a button in a blue region, and finally to a cylinder again. — $(F(\bullet \wedge X(F((\bullet \wedge \bullet) \wedge X(F\bullet)))))$
6	Navigate to a blue region, then to a button with a cylinder, and finally to a cylinder while avoiding blue regions. — $(F(\bullet \wedge X(F((\bullet \wedge \bullet) \wedge X((F\bullet) \wedge (G\neg\bullet))))))$

Table A4: Tasks in the Safety Gym domains. The RMs are generated from the LTL expressions.



Few-shot transfer

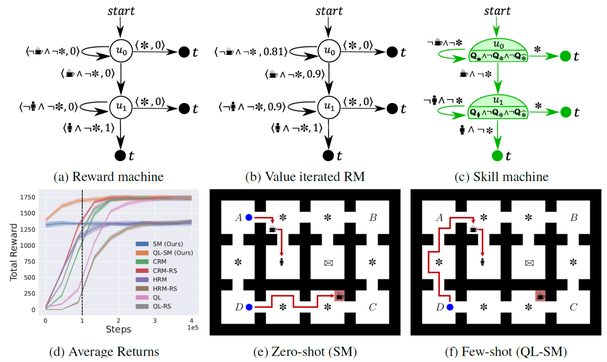
We can improve the zero-shot policy to **optimality** by using any off-policy algorithm (e.g. Q-learning)



When reachability doesn't hold

"Deliver coffee to the office and never break a decoration"

$(F(\bullet \wedge X(F\,\bullet))) \wedge (G\,\neg\bullet)$



When choosing skills greedily is bad

"Go to the office but not until you have coffee"

$(F\,\bullet) \wedge (\neg\bullet\, U\,\bullet)$