autonomous
SYSTEMS GROUP

REWARD MACHINES FOR COOPERATIVE MULTI-AGENT REINFORCEMENT LEARNING

Cyrus Neary, Zhe Xu, Bo Wu, and Ufuk Topcu
The University of Texas at Austin

AAMAS 2021 LONDON, UK
The 20th International Conference on Autonomous Agents and Multiagent Systems
3-7 May 2021, Online

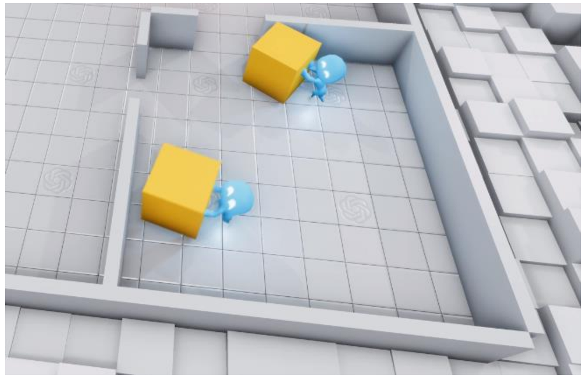
Task decomposition in multi-agent reinforcement learning

Cooperative multi-agent RL: A team of agents learn interact in a shared environment to achieve a common objective.

Task decomposition in multi-agent reinforcement learning

Cooperative multi-agent RL: A team of agents learn interact in a shared environment to achieve a common objective.

Source: Blizzard Entertainment



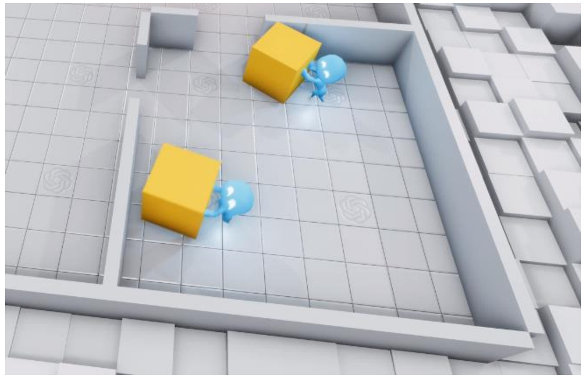
Source: OpenAI

Source: Starship Technologies

Task decomposition in multi-agent reinforcement learning

Cooperative multi-agent RL: A team of agents learn interact in a shared environment to achieve a common objective.


Source: Blizzard Entertainment

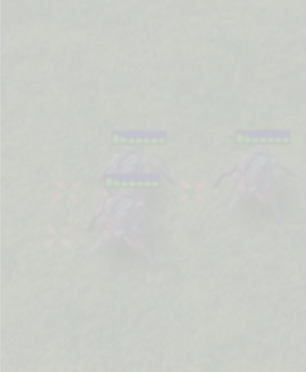

Source: OpenAI


Source: Starship Technologies

Observation: Often, agents only interact in several crucial moments of the task.

Task decomposition in multi-agent reinforcement learning

Cooperative multi-agent RL: A team of agents learn interact in a shared environment to achieve a common objective.



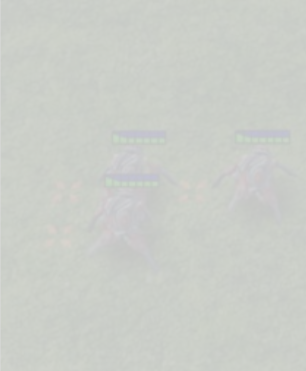
Objectives:

1) Present a framework for specifying structured representations of team tasks.

Observation: Often, agents only interact in several crucial moments of the task.

Task decomposition in multi-agent reinforcement learning

Cooperative multi-agent RL: A team of agents learn interact in a shared environment to achieve a common objective.



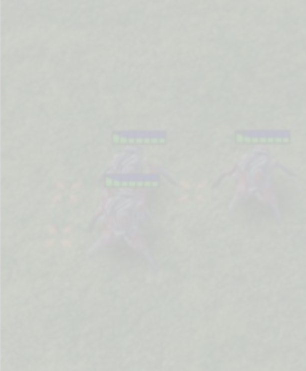
Objectives:

1) Present a framework for specifying structured representations of team tasks.
2) Use this specification to decompose problem into necessary individual behaviors.

Observation: Often, agents only interact in several crucial moments of the task.

Task decomposition in multi-agent reinforcement learning

Cooperative multi-agent RL: A team of agents learn interact in a shared environment to achieve a common objective.

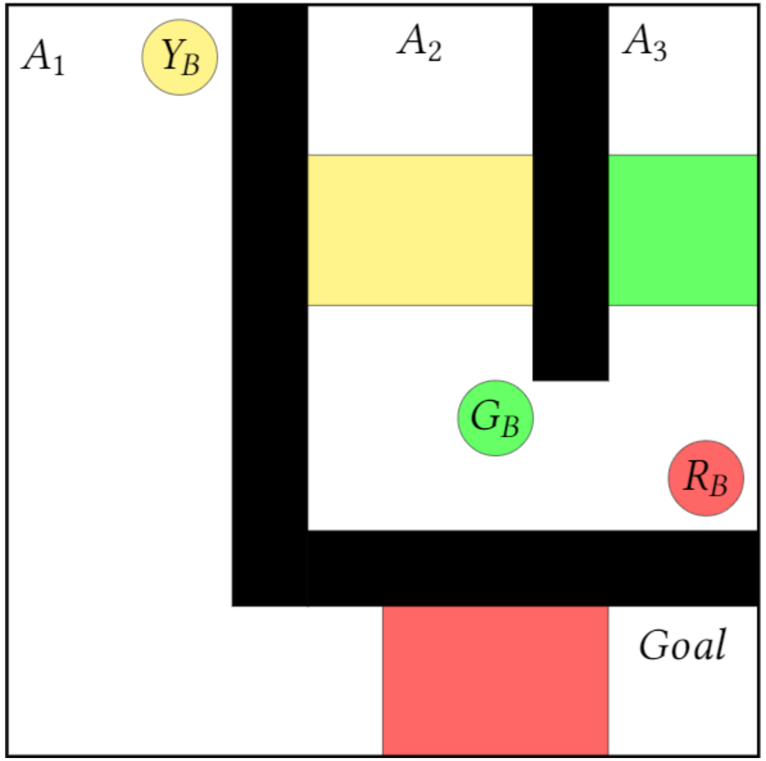


Objectives:

1) Present a framework for specifying structured representations of team tasks.
2) Use this specification to decompose problem into necessary individual behaviors.
3) Present a reinforcement learning algorithm that uses decomposition to simplify multi-agent learning.

Observation: Often, agents only interact in several crucial moments of the task.

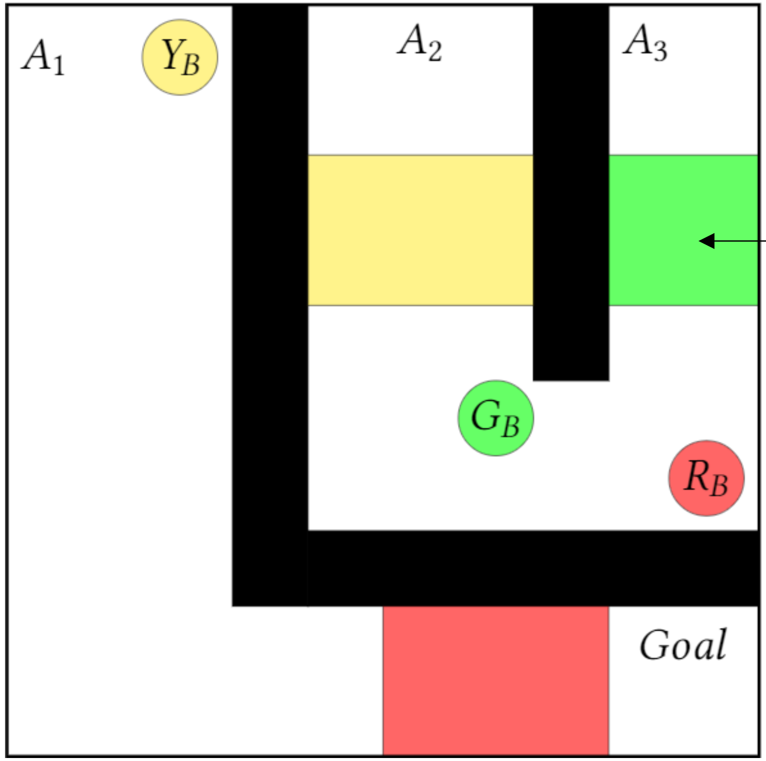
An Illustrative Running Example



Team objective:

Have A_1 safely reach the goal location $Goal$.

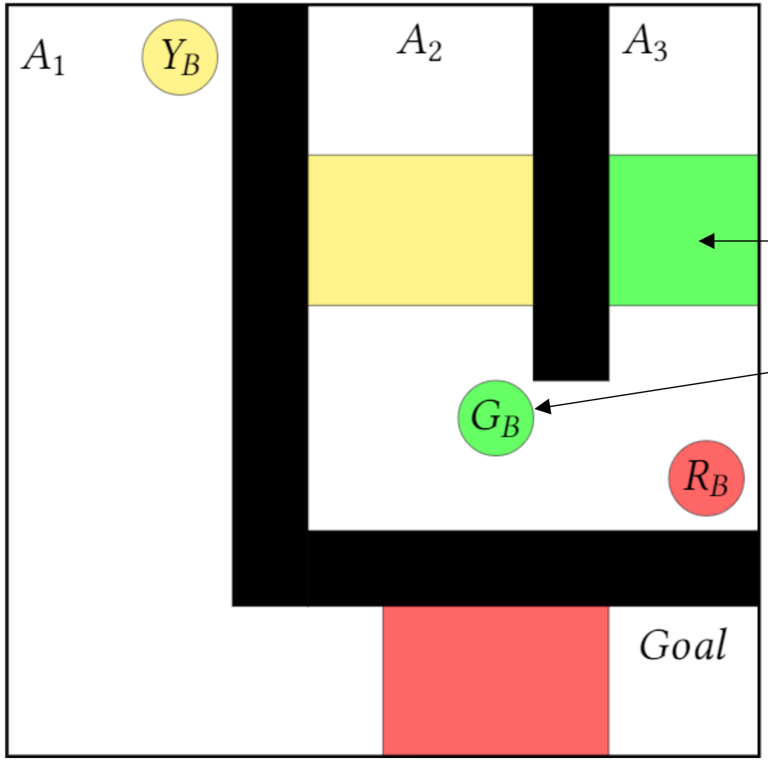
An Illustrative Running Example



Team objective:
Have A_1 safely reach the goal location $Goal$.

Colored walls block the agents' paths.

An Illustrative Running Example

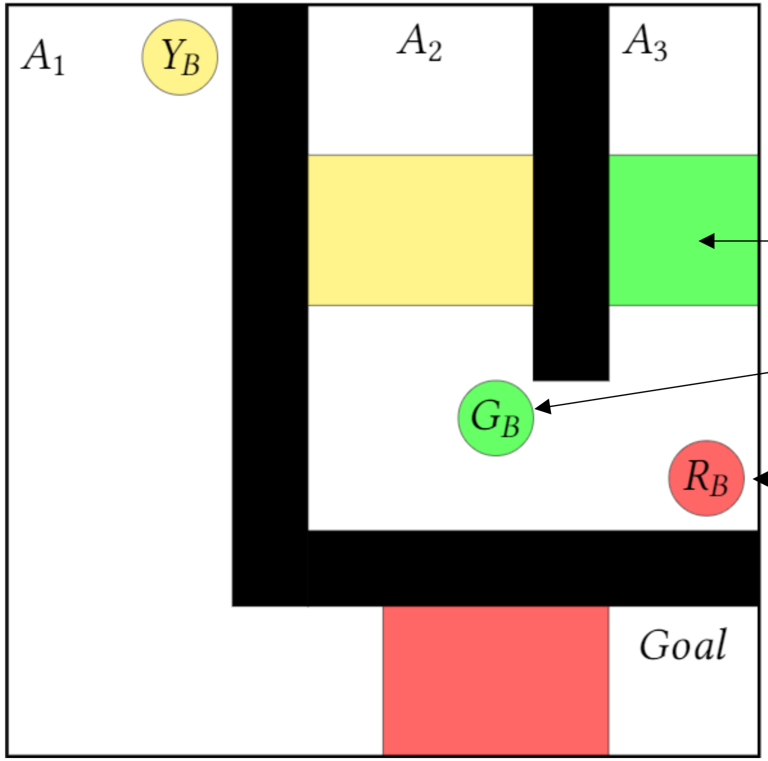


Team objective:
Have A_1 safely reach the goal location $Goal$.

Colored walls block the agents' paths.

Buttons remove the wall of the corresponding color.

An Illustrative Running Example



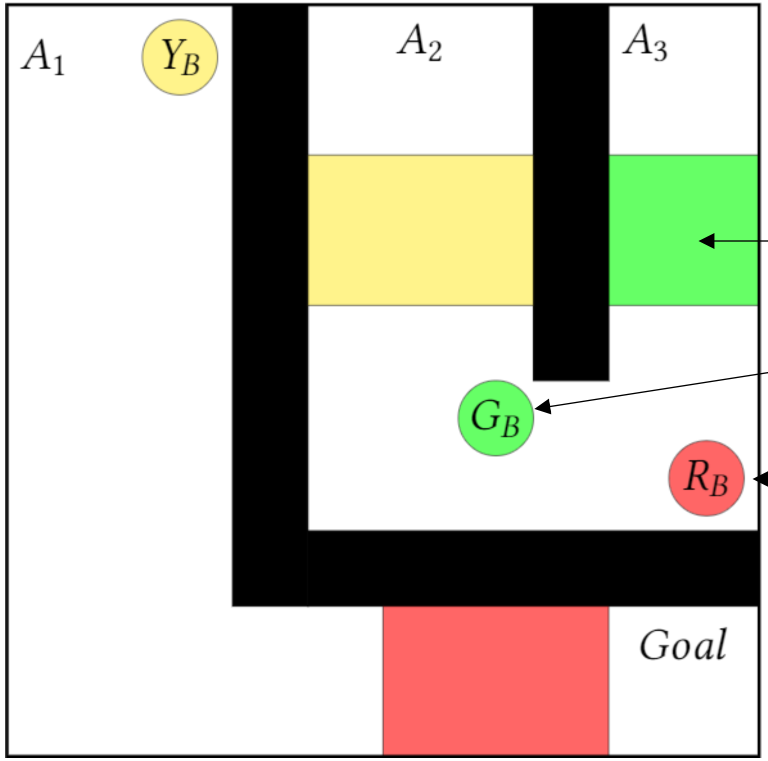
Team objective:
Have A_1 safely reach the goal location $Goal$.

Colored walls block the agents' paths.

Buttons remove the wall of the corresponding color.

Pressing the red button requires two agents.

An Illustrative Running Example



Team objective:
Have A_1 safely reach the goal location $Goal$.

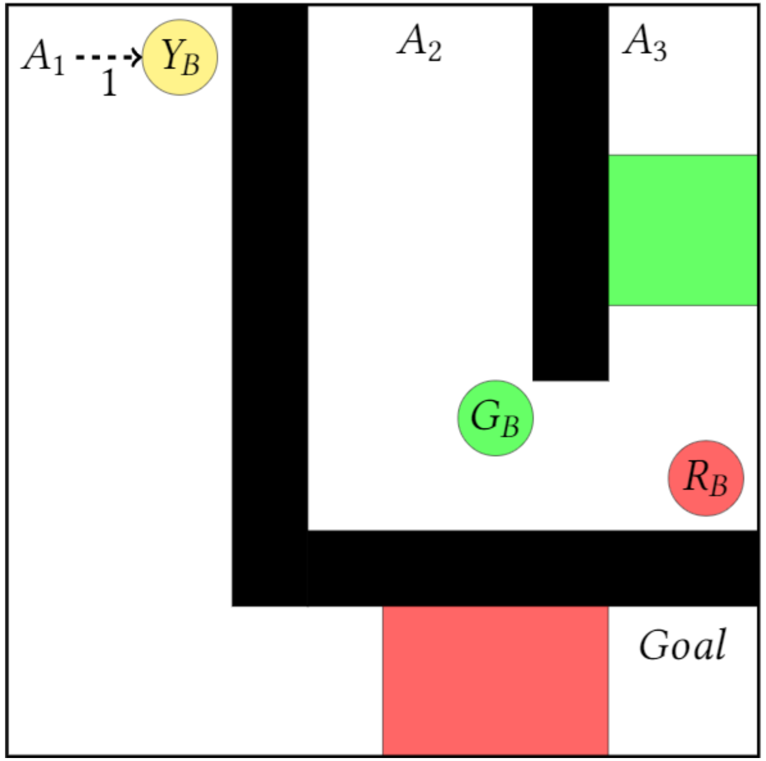
Colored walls block the agents' paths.

Buttons remove the wall of the corresponding color.

Pressing the red button requires two agents.

- Environment states $\overbrace{\boldsymbol{S}}^{\text{Set of team environment states}} = S_1 \times S_2 \times \overbrace{S_3}^{\text{Local states of agent 3}}$
- Environment actions $\underbrace{\boldsymbol{A}}_{\text{Set of team action}} = A_1 \times A_2 \times \underbrace{A_3}_{\text{Actions available to agent 3}}$

An Illustrative Running Example



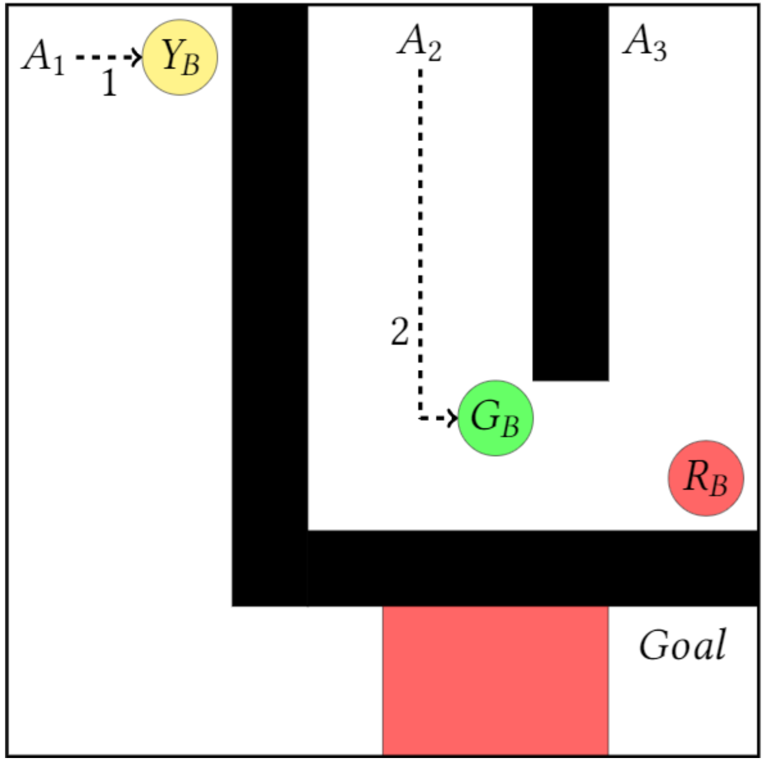
Team objective:
Have A_1 safely reach the goal location $Goal.$

Colored walls block the agents' paths.

Buttons remove the wall of the corresponding color.

Pressing the red button requires two agents.

An Illustrative Running Example



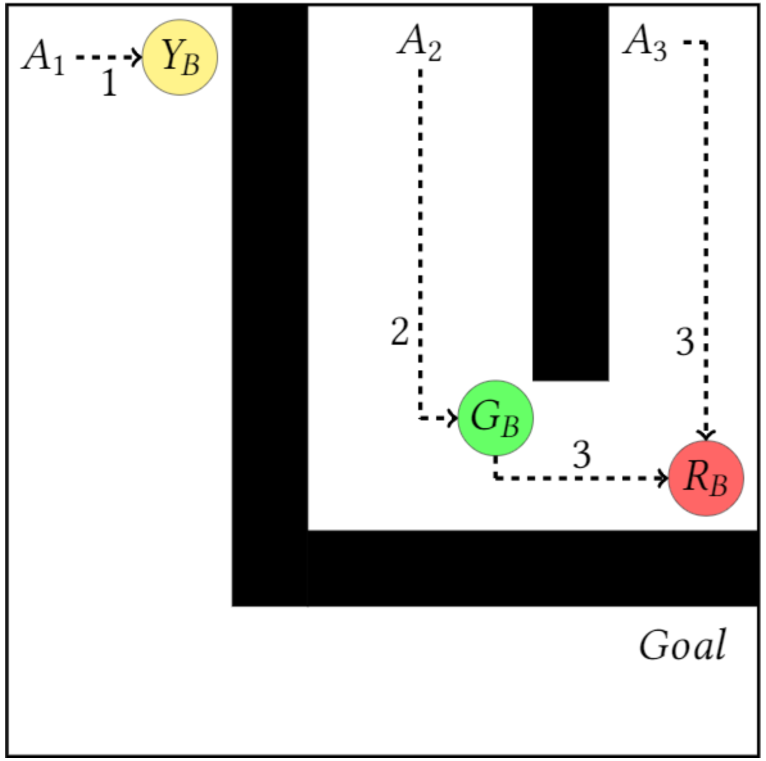
Team objective:
Have A_1 safely reach the goal location $Goal$.

Colored walls block the agents' paths.

Buttons remove the wall of the corresponding color.

Pressing the red button requires two agents.

An Illustrative Running Example



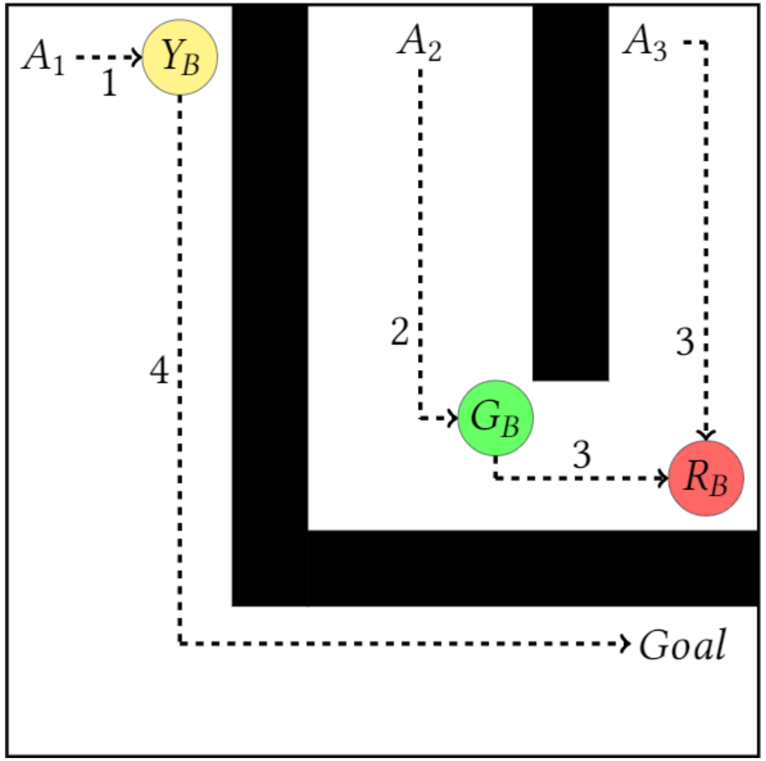
Team objective:
Have A_1 safely reach the goal location $Goal$.

Colored walls block the agents' paths.

Buttons remove the wall of the corresponding color.

Pressing the red button requires two agents.

An Illustrative Running Example



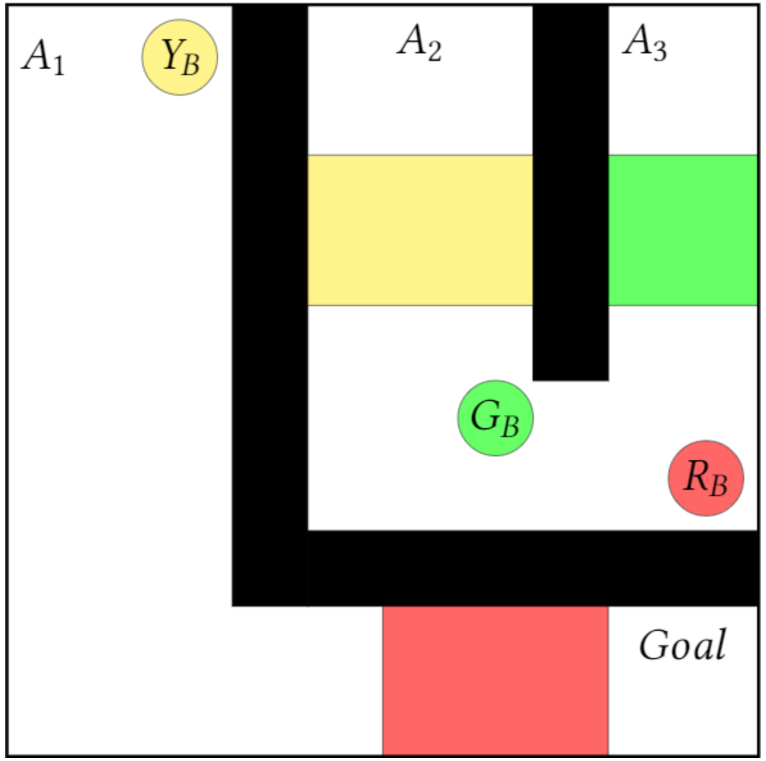
Team objective:
Have A_1 safely reach the goal location $Goal$.

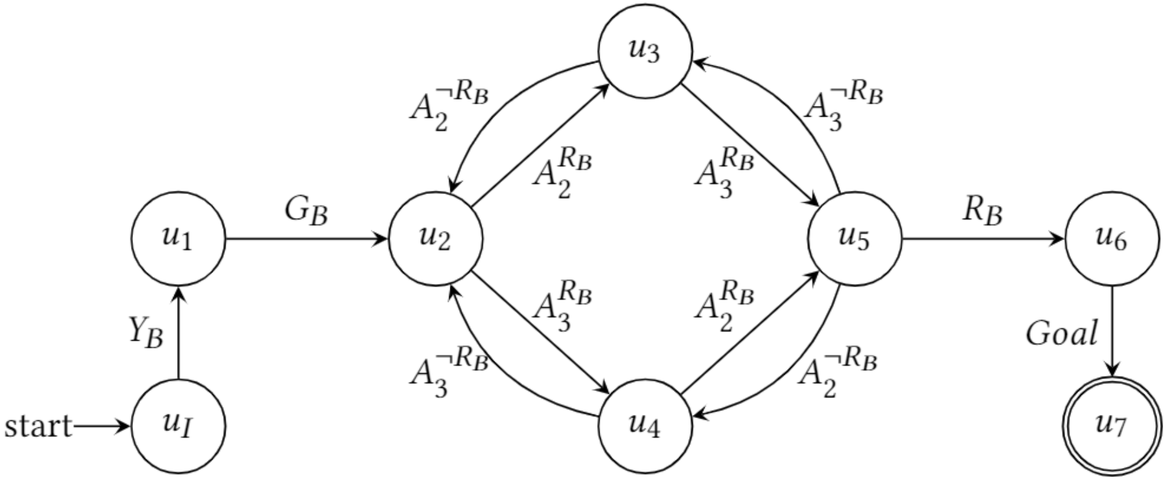
Colored walls block the agents' paths.

Buttons remove the wall of the corresponding color.

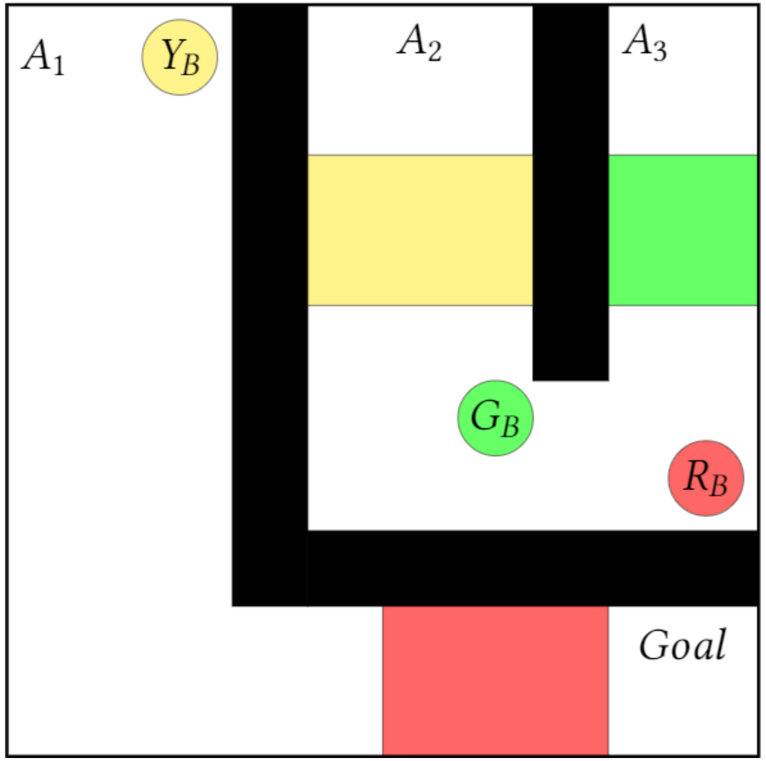
Pressing the red button requires two agents.

Representing Temporally Extended Team Tasks – Reward Machines

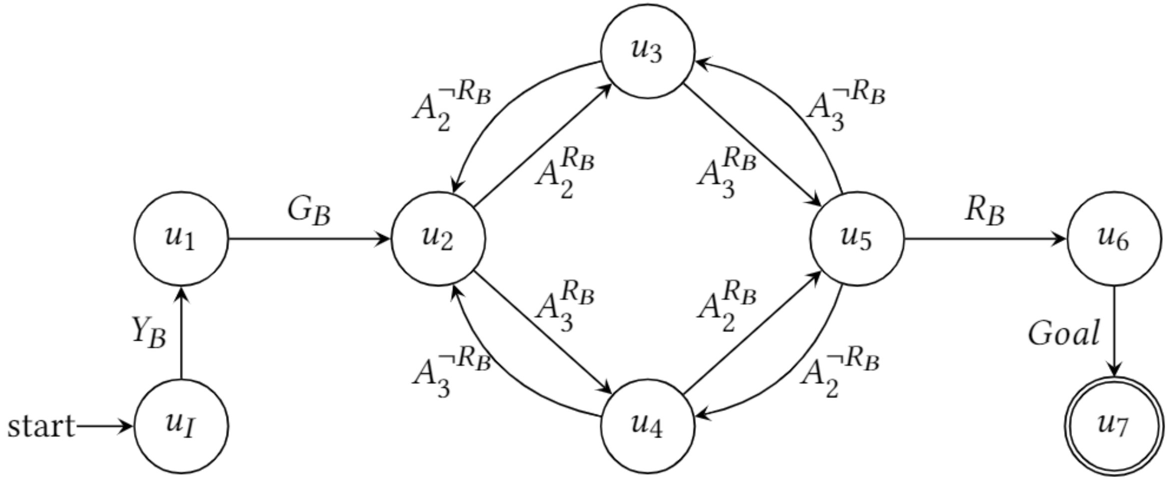




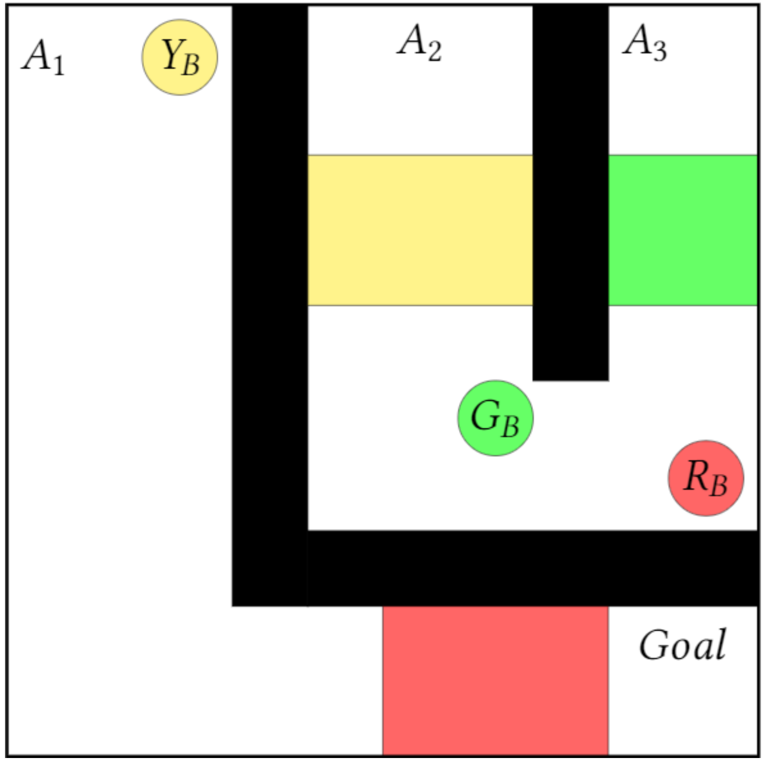
Representing Temporally Extended Team Tasks – Reward Machines



$U = \{u_I, u_1, u_2, \ldots, u_7\}$ – Set of states

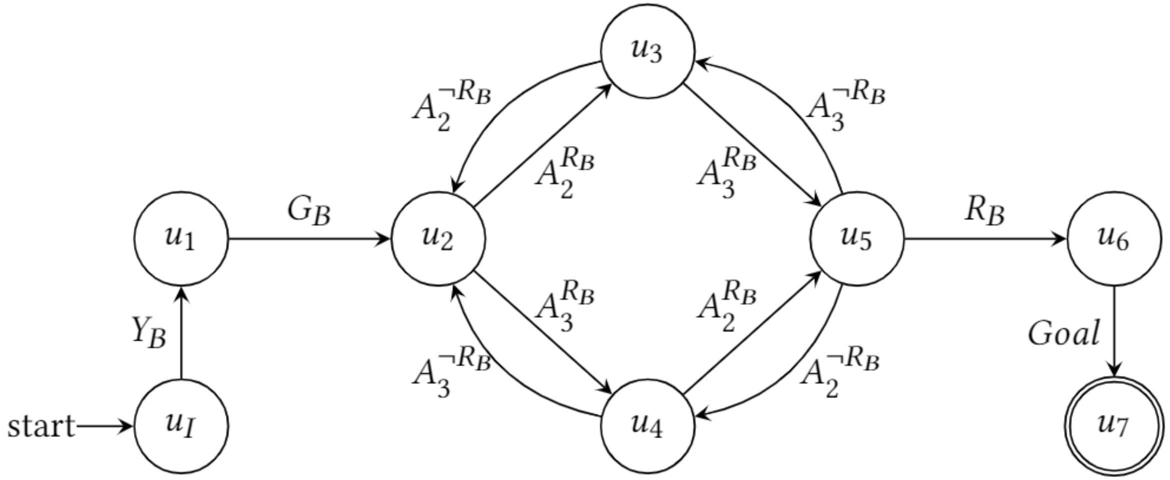


Representing Temporally Extended Team Tasks – Reward Machines

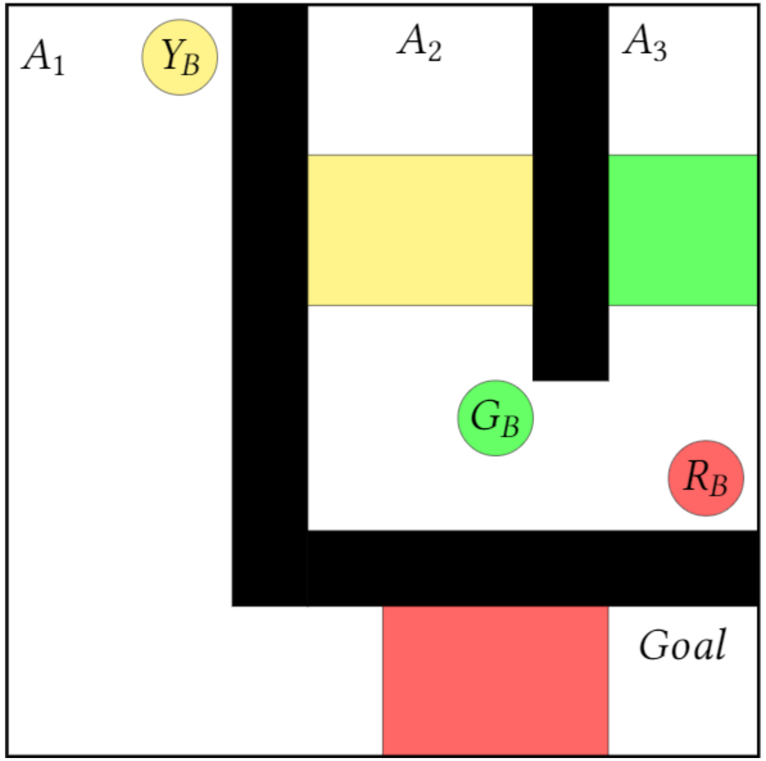


$U = \{u_I, u_1, u_2, \ldots, u_7\}$ – Set of states

$\Sigma = \{Y_B, G_B, R_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, Goal\}$ – Set of events



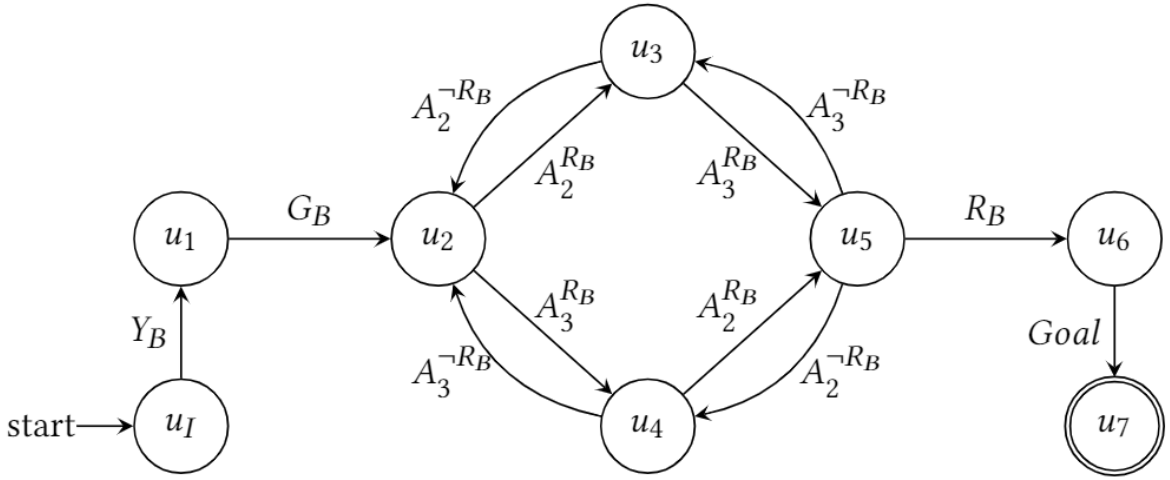
Representing Temporally Extended Team Tasks – Reward Machines



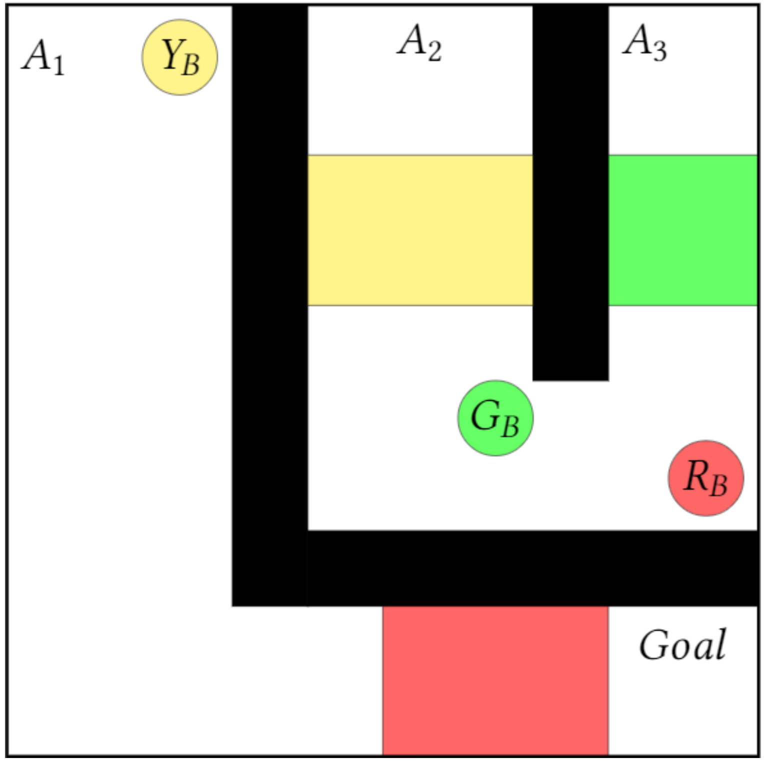
$U = \{u_I, u_1, u_2, \ldots, u_7\}$ – Set of states

$\Sigma = \{Y_B, G_B, R_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, Goal\}$ – Set of events

$\delta : U \times \Sigma \rightarrow U$ – Transition function



Representing Temporally Extended Team Tasks – Reward Machines



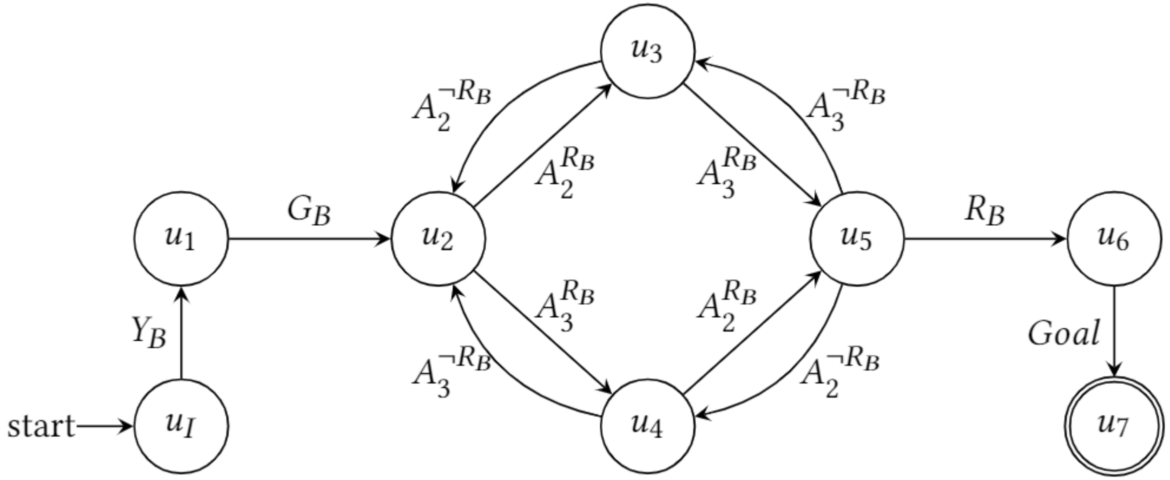
$U = \{u_I, u_1, u_2, \dots, u_7\}$ – Set of states

$\Sigma = \{Y_B, G_B, R_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, Goal\}$ – Set of events

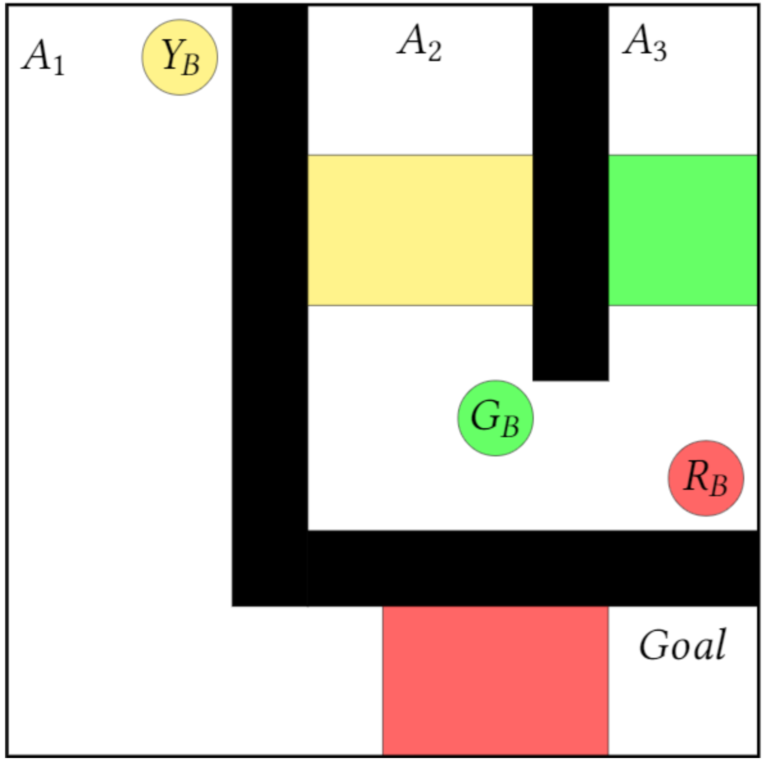
$\delta : U \times \Sigma \to U$ – Transition function

$\sigma : U \times U \to \mathbb{R}$ – Output function

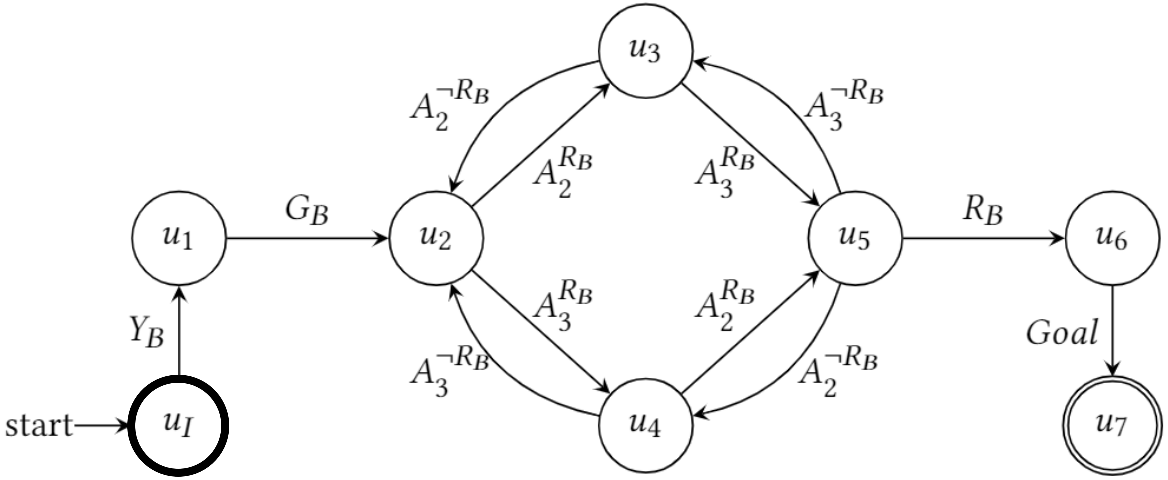
F – Set of final states



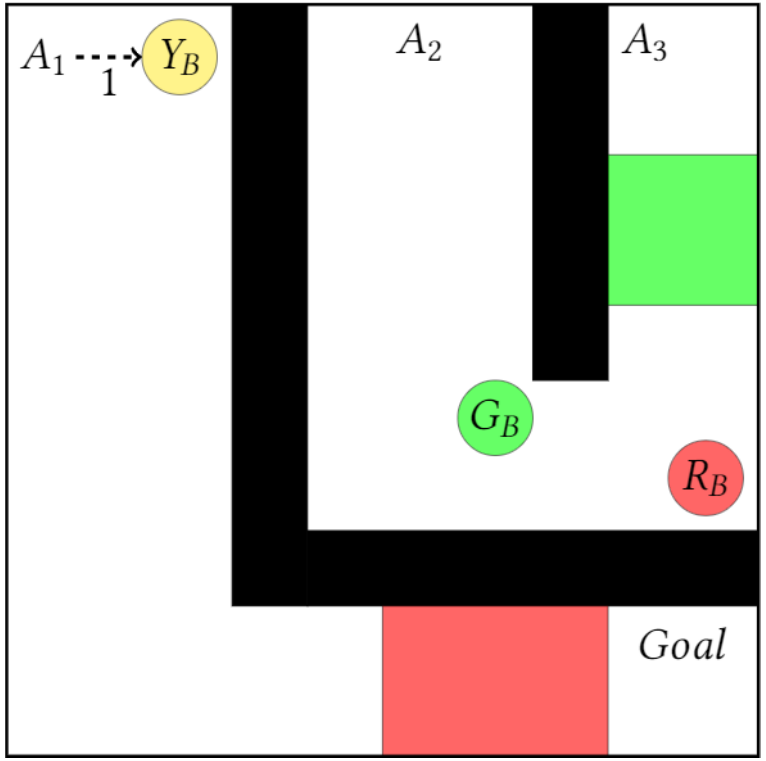
Representing Temporally Extended Team Tasks – Reward Machines



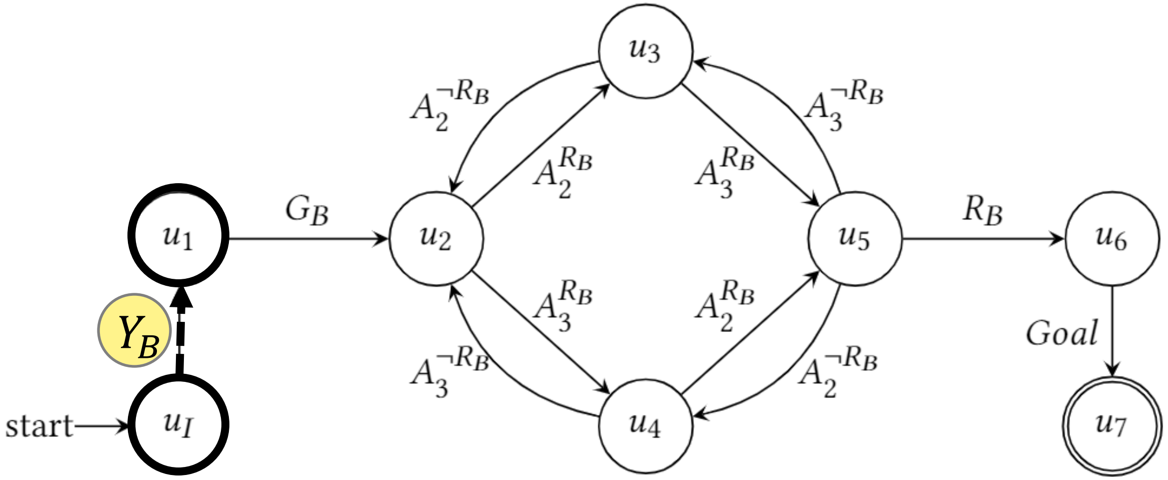
$$\Sigma = \{Y_B, G_B, R_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, Goal\}$$



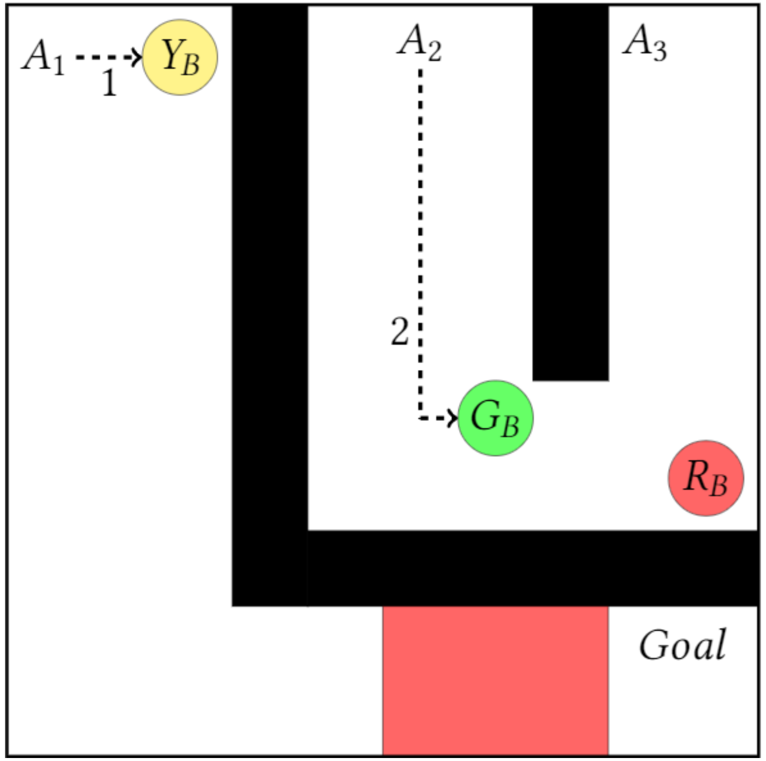
Representing Temporally Extended Team Tasks – Reward Machines



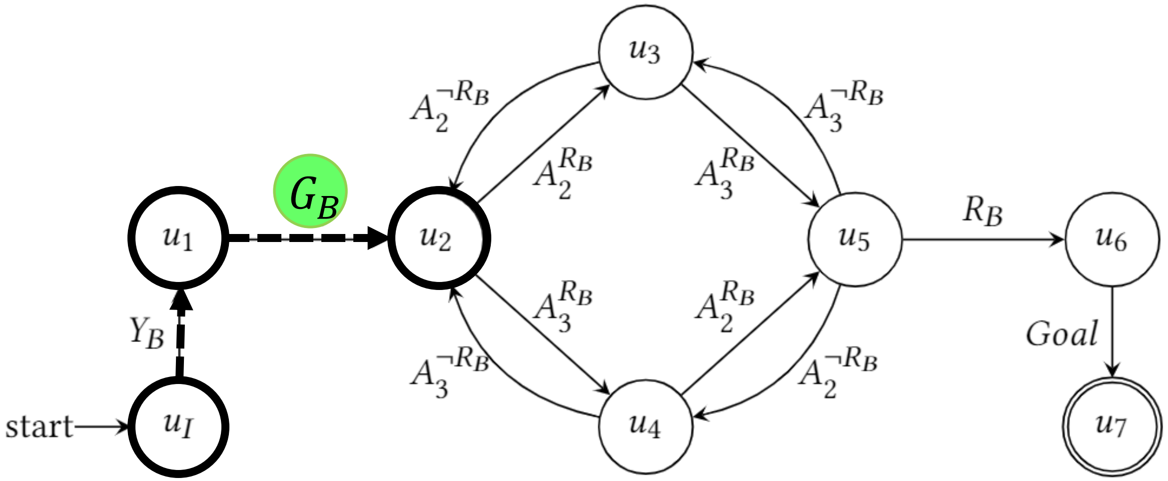
$$\Sigma = \{Y_B, G_B, R_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, Goal\}$$



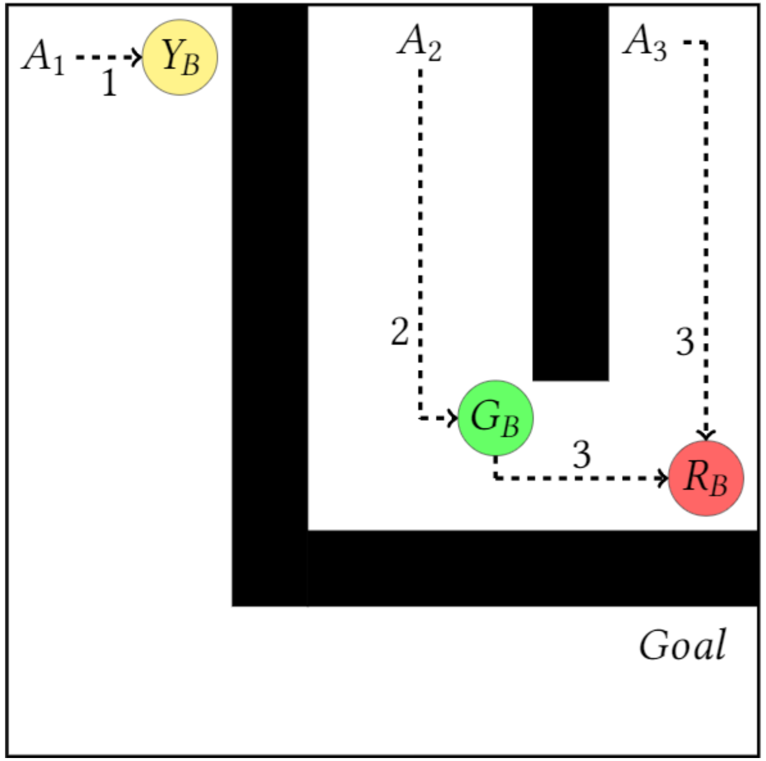
Representing Temporally Extended Team Tasks – Reward Machines



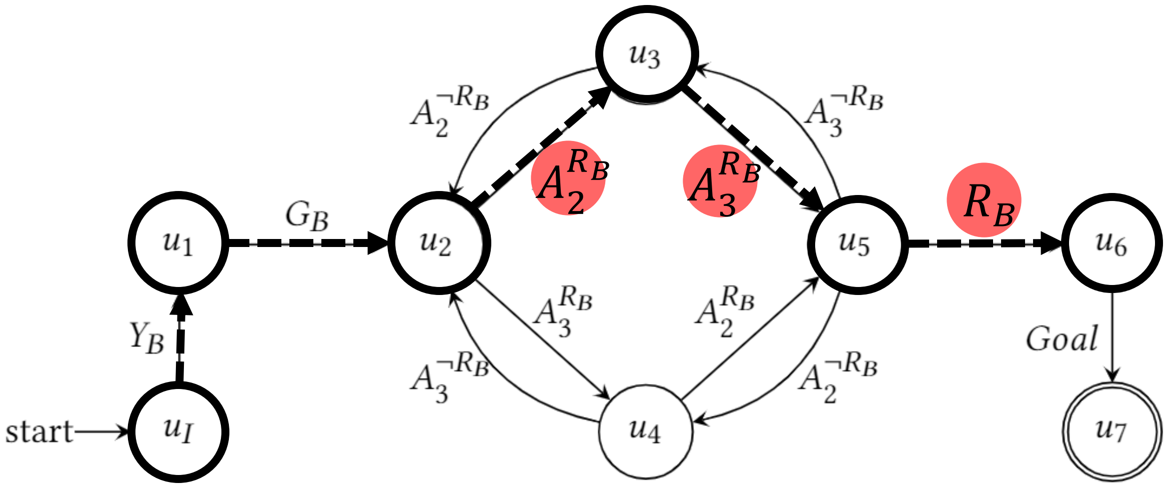
$$\Sigma = \{Y_B, G_B, R_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, Goal\}$$



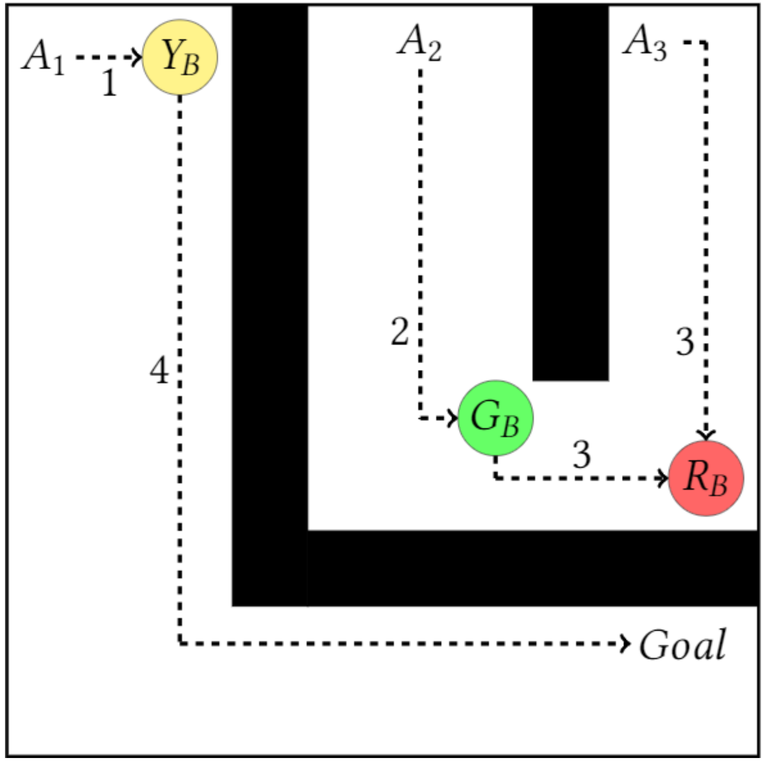
Representing Temporally Extended Team Tasks – Reward Machines



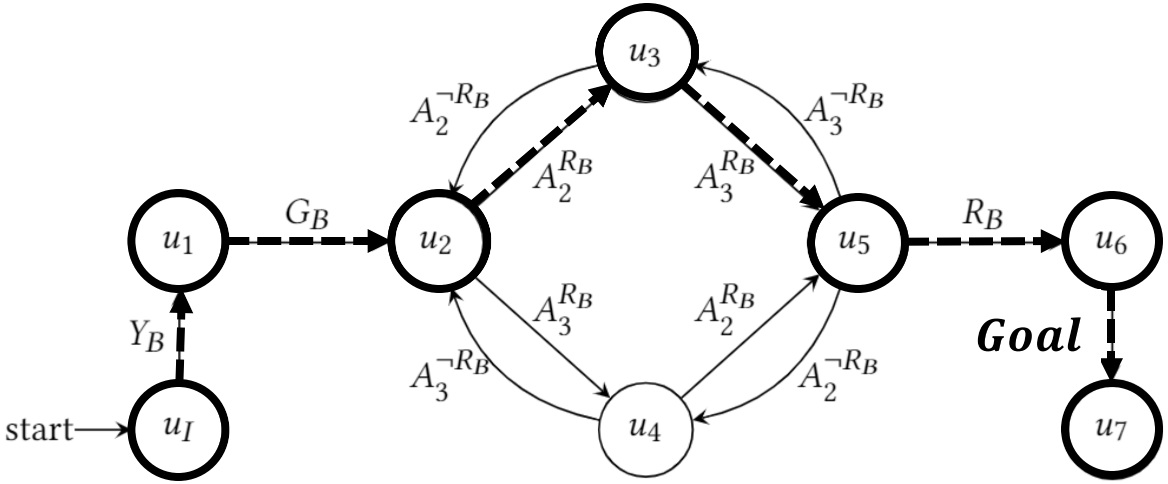
$$\Sigma = \{Y_B, G_B, R_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, Goal\}$$



Representing Temporally Extended Team Tasks – Reward Machines

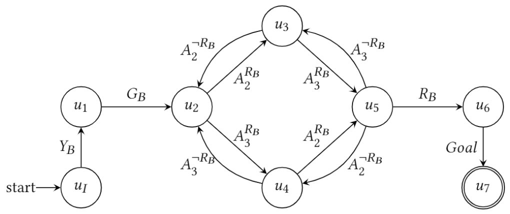


$$\Sigma = \{Y_B, G_B, R_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, \boldsymbol{Goal}\}$$



Connecting Environment States to a Reward Machine

The labeling function: Relate the environment state to collections of high-level events.

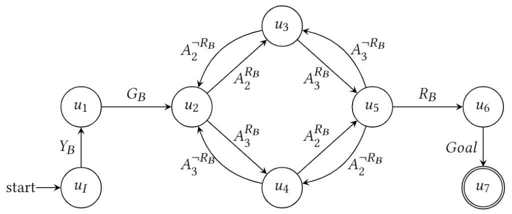


High-Level Task
(Model: Reward Machine)

- Reward machine states U
- High-level Events Σ

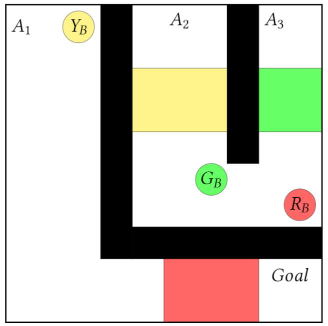
Connecting Environment States to a Reward Machine

The labeling function: Relate the environment state to collections of high-level events.



High-Level Task
(Model: Reward Machine)

- Reward machine states U
- High-level Events Σ

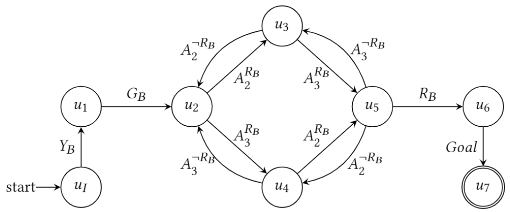


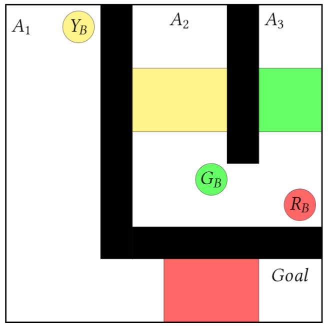
Environment Dynamics Model: Transition distribution $p(\cdot\,|\boldsymbol{s}, \boldsymbol{a})$

- Environment states $\boldsymbol{S} = S_1 \times S_2 \times S_3$
- Environment actions $\boldsymbol{A} = A_1 \times A_2 \times A_3$

Connecting Environment States to a Reward Machine

The labeling function: Relate the environment state to collections of high-level events.





High-Level Task
(Model: Reward Machine)

Events from Σ

Labeling
function L

Environment states

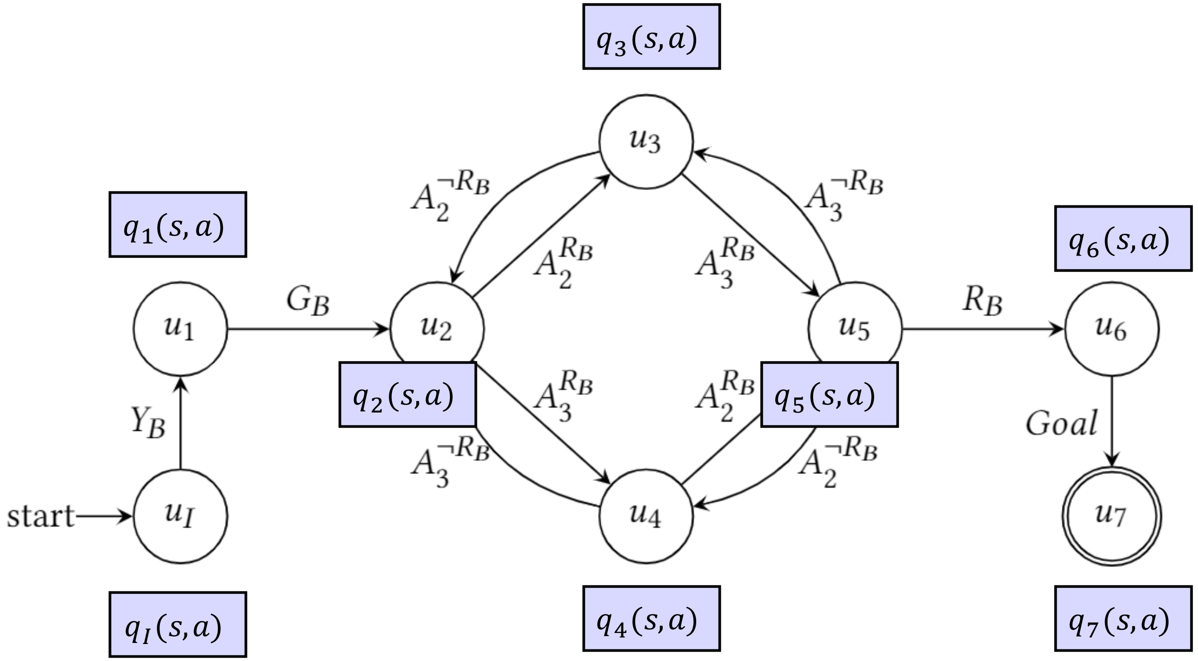
Environment Dynamics
Model: Transition
distribution $p(\cdot\,|\boldsymbol{s}, \boldsymbol{a})$

- Reward machine states U
- High-level Events Σ

- Environment states $\boldsymbol{S} = S_1 \times S_2 \times S_3$
- Environment actions $\boldsymbol{A} = A_1 \times A_2 \times A_3$

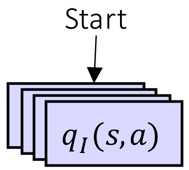
Using Reward Machines for Reinforcement Learning

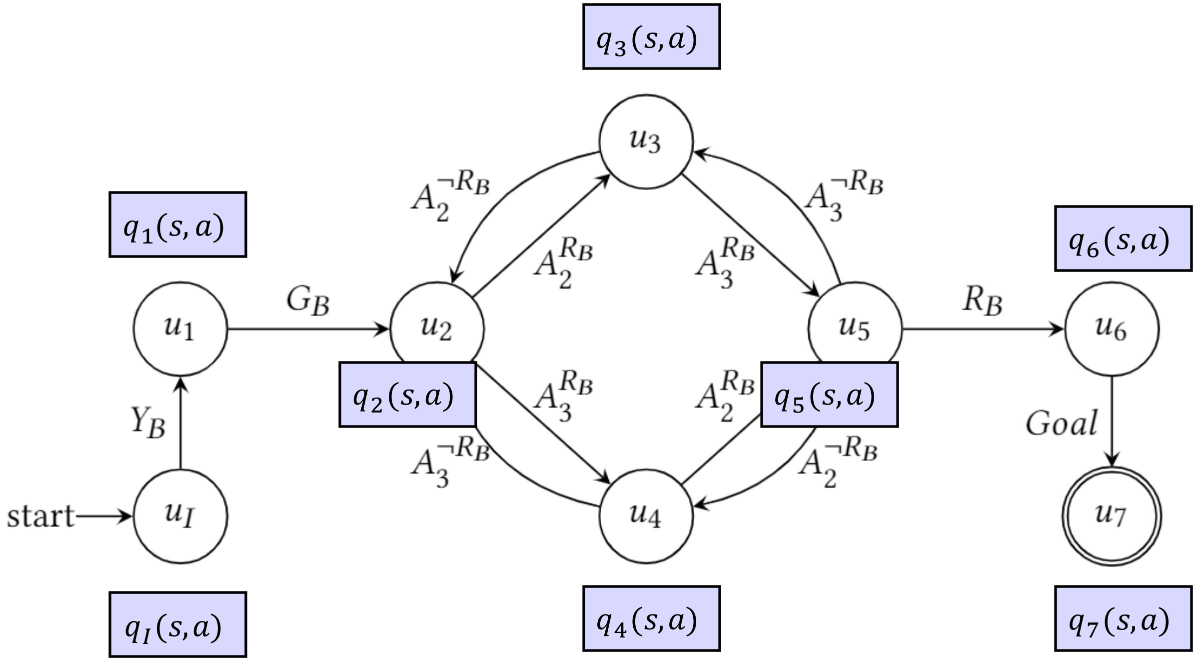
Perform q-learning to learn a separate policy for each state of the reward machine.



Using Reward Machines for Reinforcement Learning

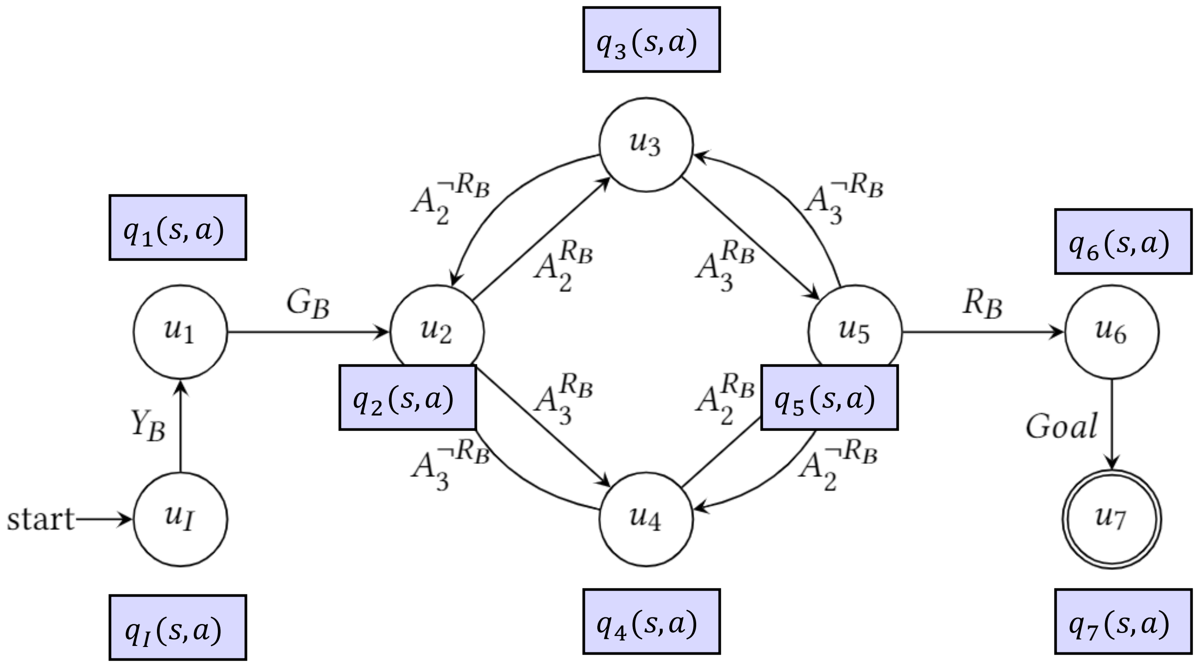
Perform q-learning to learn a separate policy for each state of the reward machine.

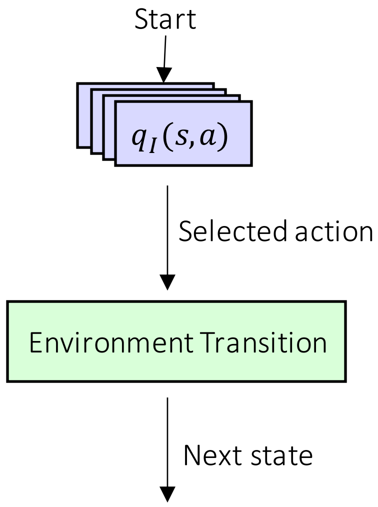




Using Reward Machines for Reinforcement Learning

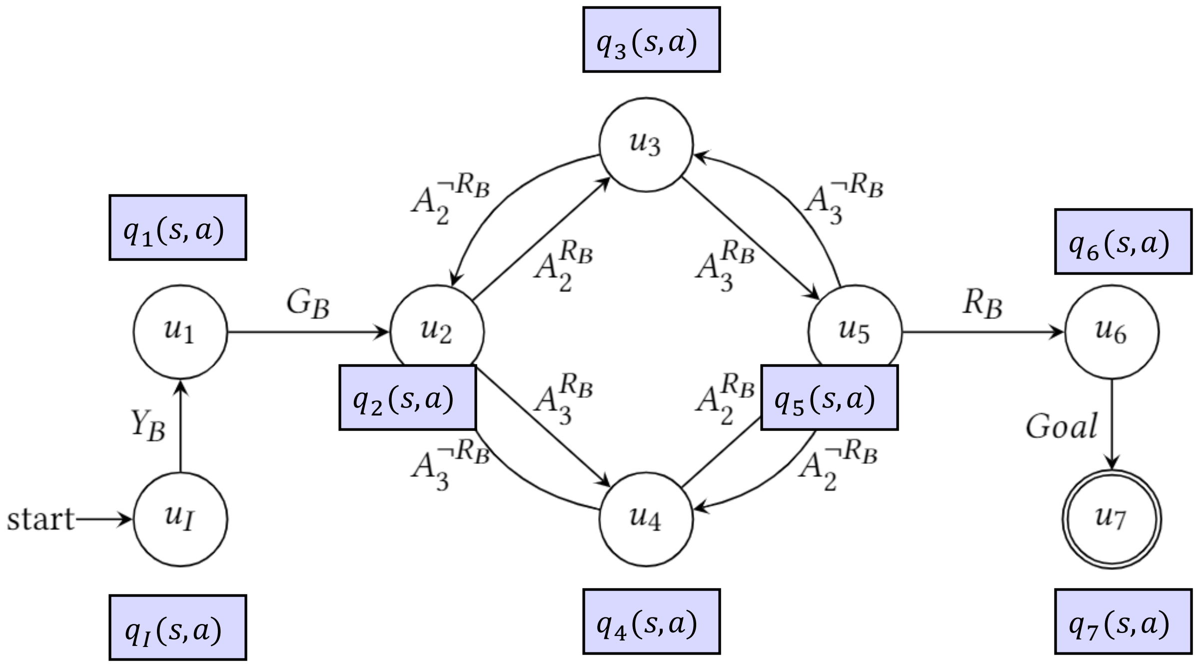
Perform q-learning to learn a separate policy for each state of the reward machine.

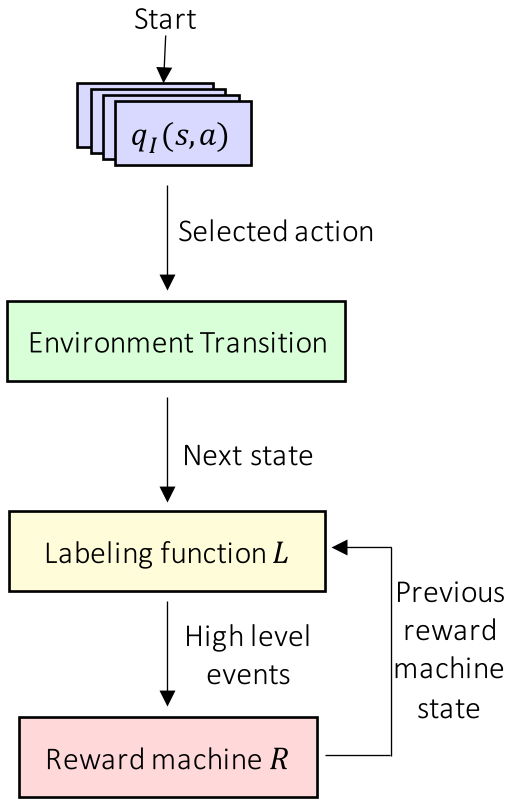




Using Reward Machines for Reinforcement Learning

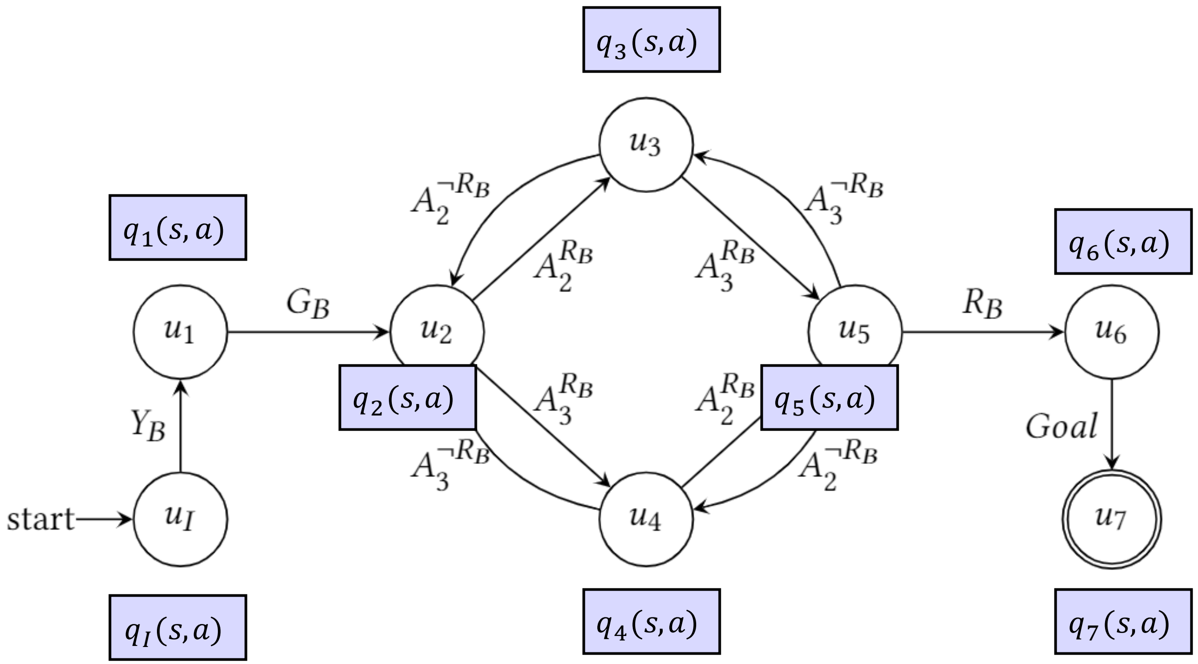
Perform q-learning to learn a separate policy for each state of the reward machine.

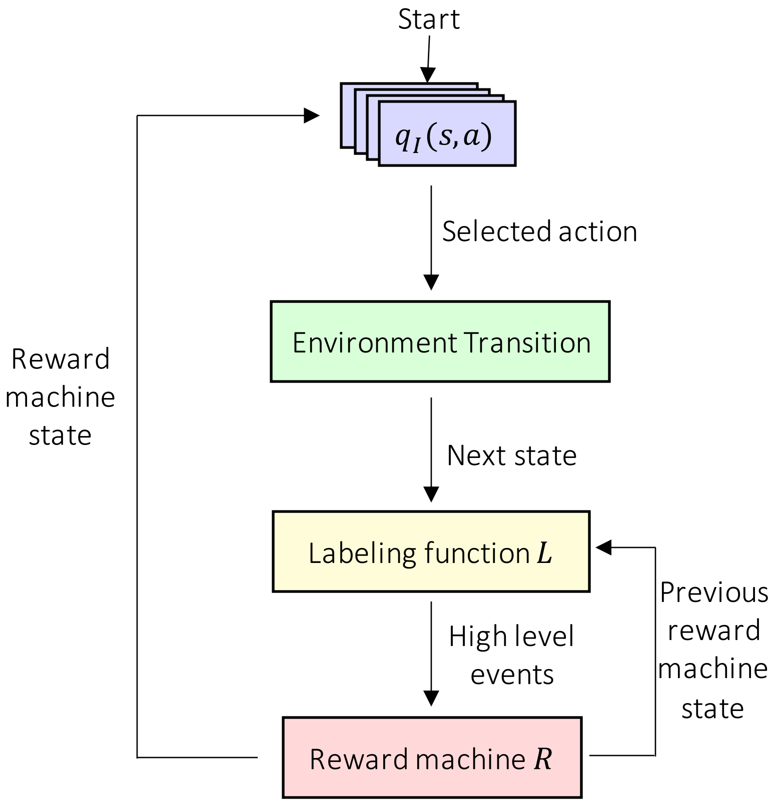




Using Reward Machines for Reinforcement Learning

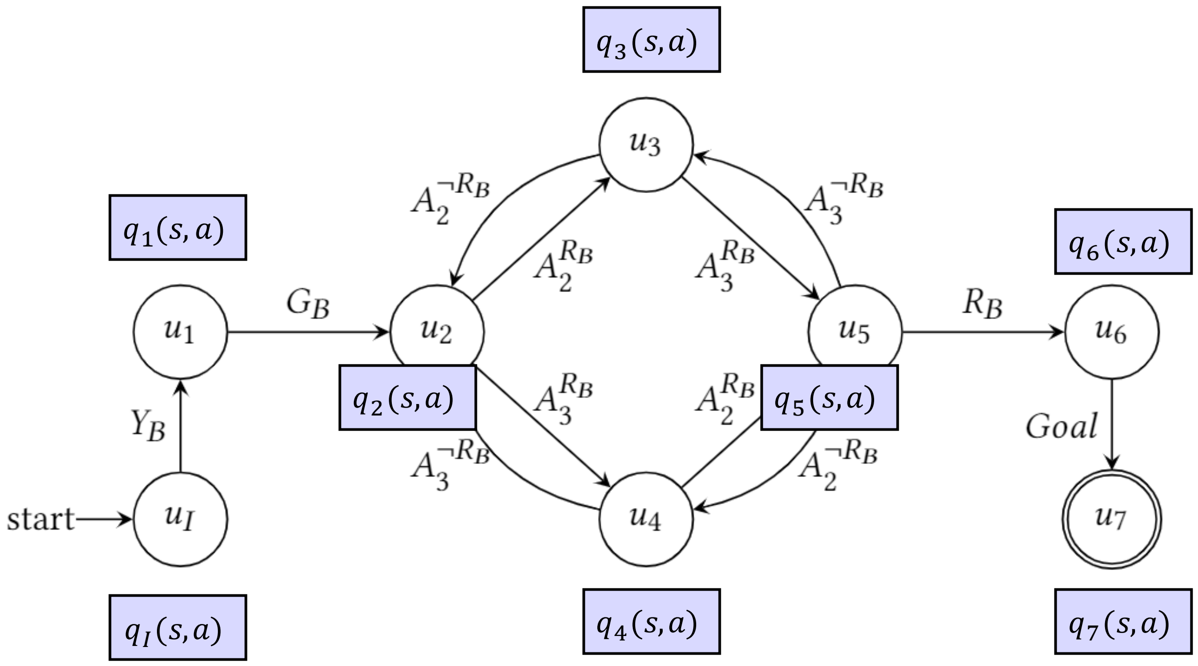
Perform q-learning to learn a separate policy for each state of the reward machine.

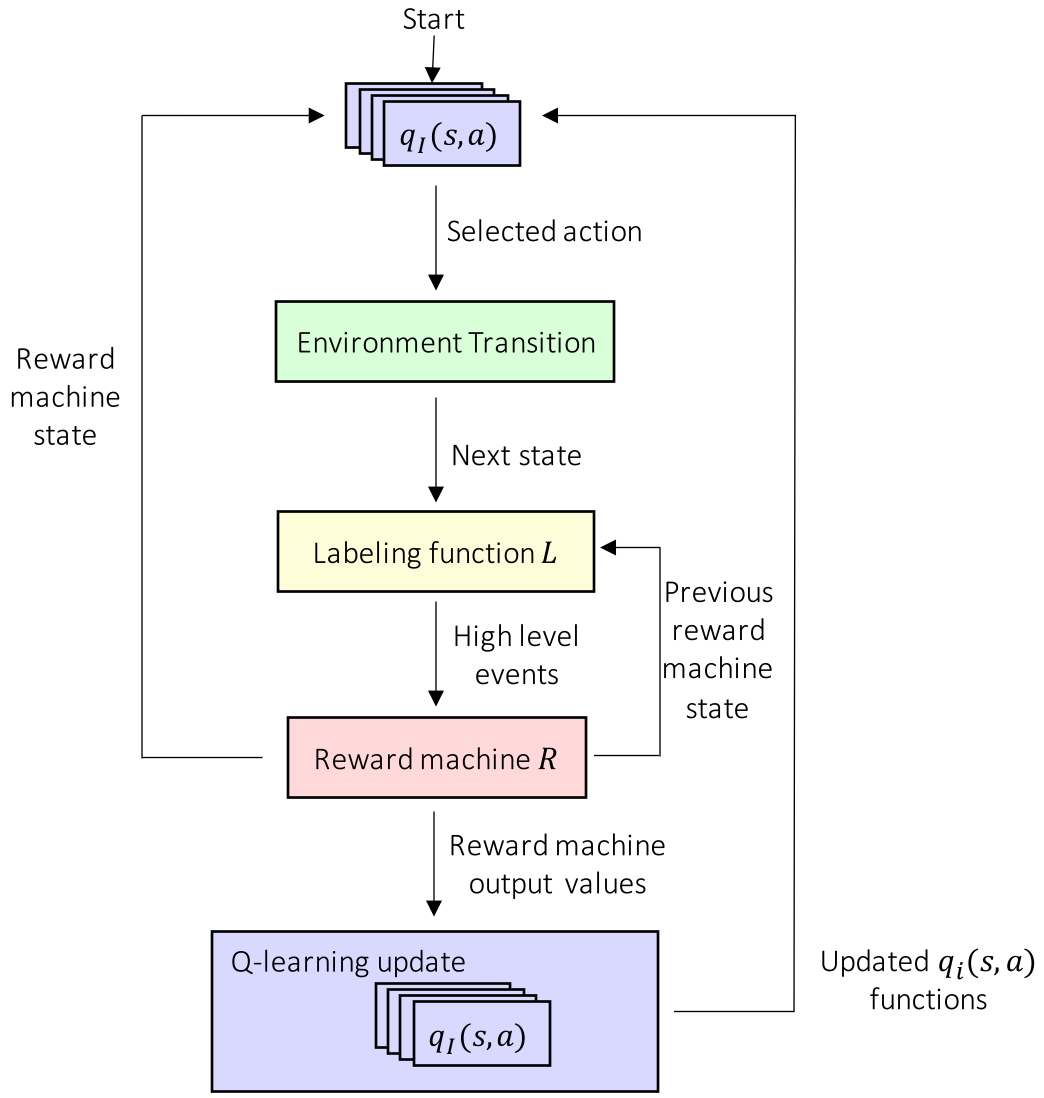




Using Reward Machines for Reinforcement Learning

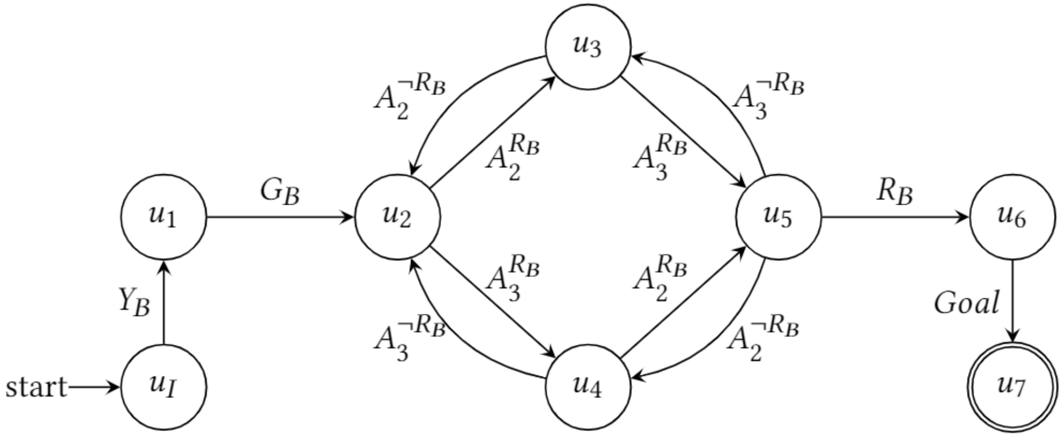
Perform q-learning to learn a separate policy for each state of the reward machine.





Reward Machine Projection

How does the reward machine change if one only has access to events from $\Sigma_i \subseteq \Sigma$?

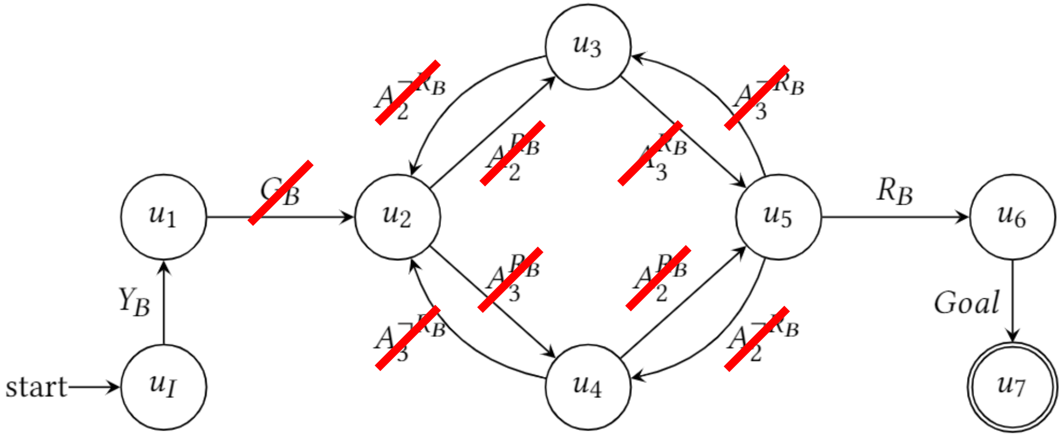


Example: Event subset for A_1 is

$$\Sigma_1 = \{Y_B, R_B, Goal\}$$

Reward Machine Projection

How does the reward machine change if one only has access to events from $\Sigma_i \subseteq \Sigma$?



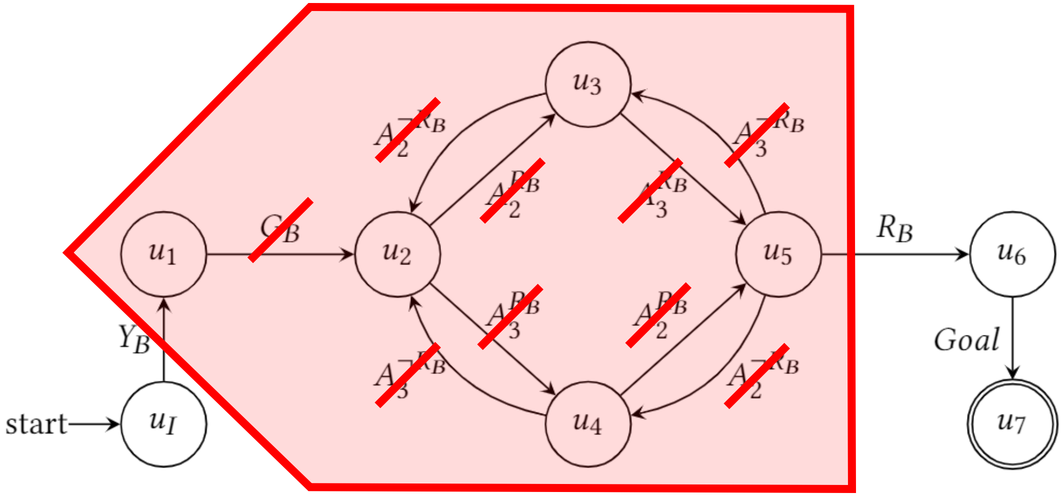
Example: Event subset for A_1 is

$$\Sigma_1 = \{Y_B, R_B, Goal\}$$

- Transitions triggered by missing events are not observed.

Reward Machine Projection

How does the reward machine change if one only has access to events from $\Sigma_i \subseteq \Sigma$?



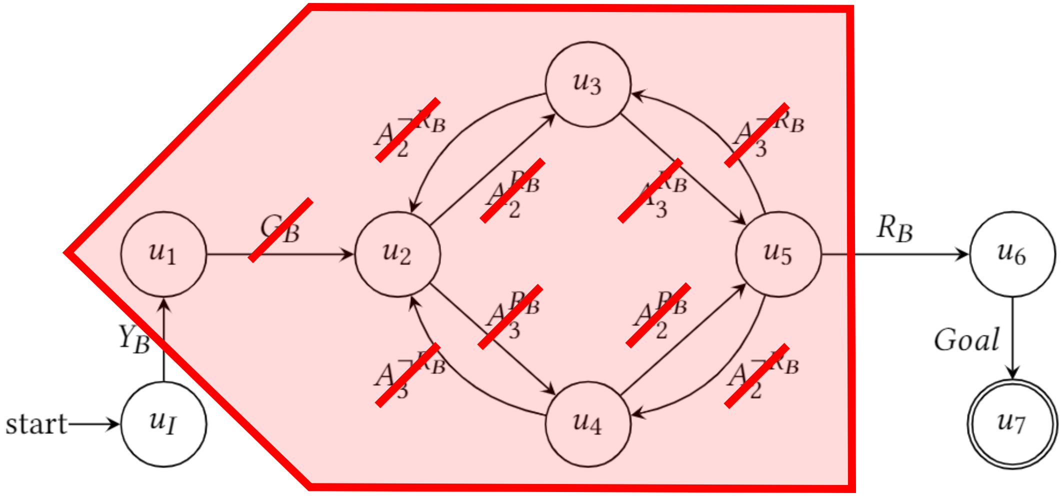
Example: Event subset for A_1 is

$$\Sigma_1 = \{Y_B, R_B, Goal\}$$

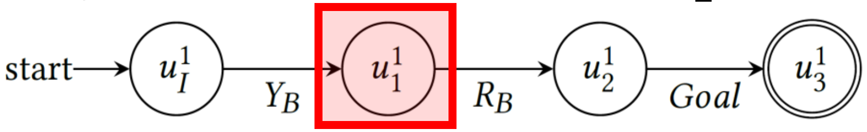
- Transitions triggered by missing events are not observed.
- Merge states that cannot be differentiated by events from Σ_1.

Reward Machine Projection

How does the reward machine change if one only has access to events from $\Sigma_i \subseteq \Sigma$?



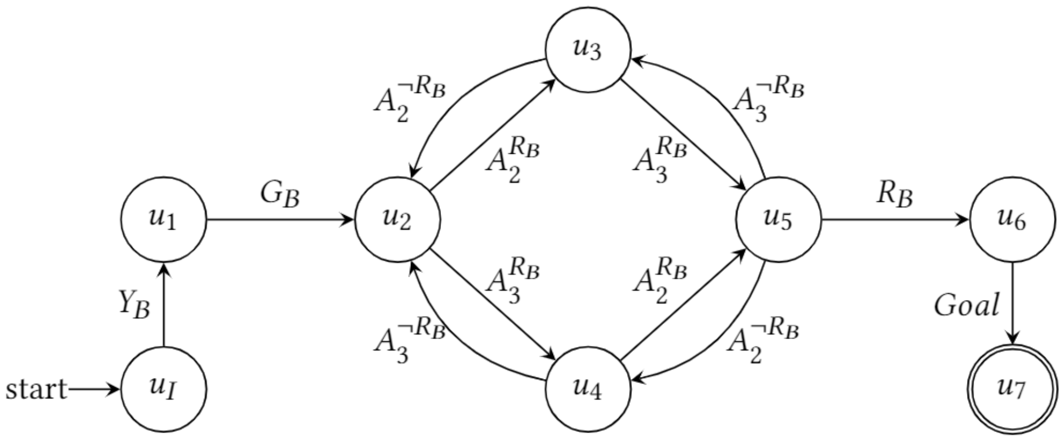
Projected Reward Machine R_1



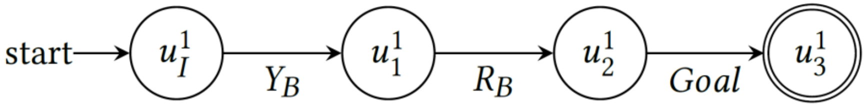
- Transitions triggered by missing events are not observed.
- Merge states that cannot be differentiated by events from Σ_1.

Reward Machine Projection

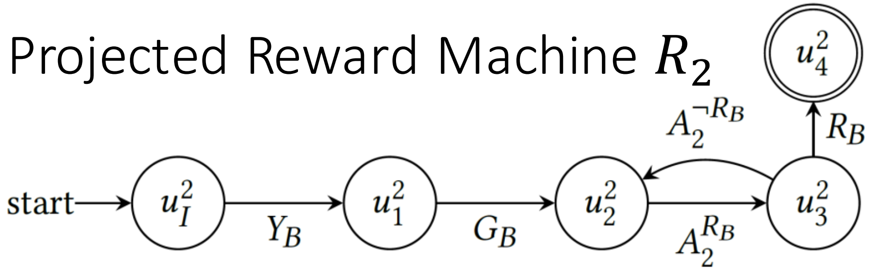
How does the reward machine change if one only has access to events from $\Sigma_i \subseteq \Sigma$?



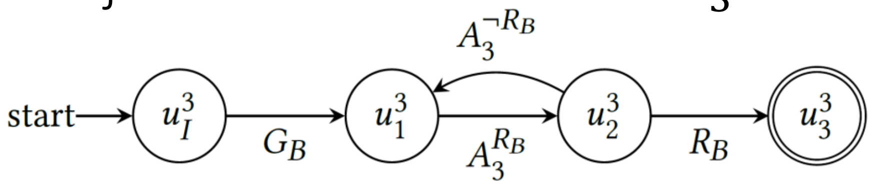
Projected Reward Machine R_1



Projected Reward Machine R_2



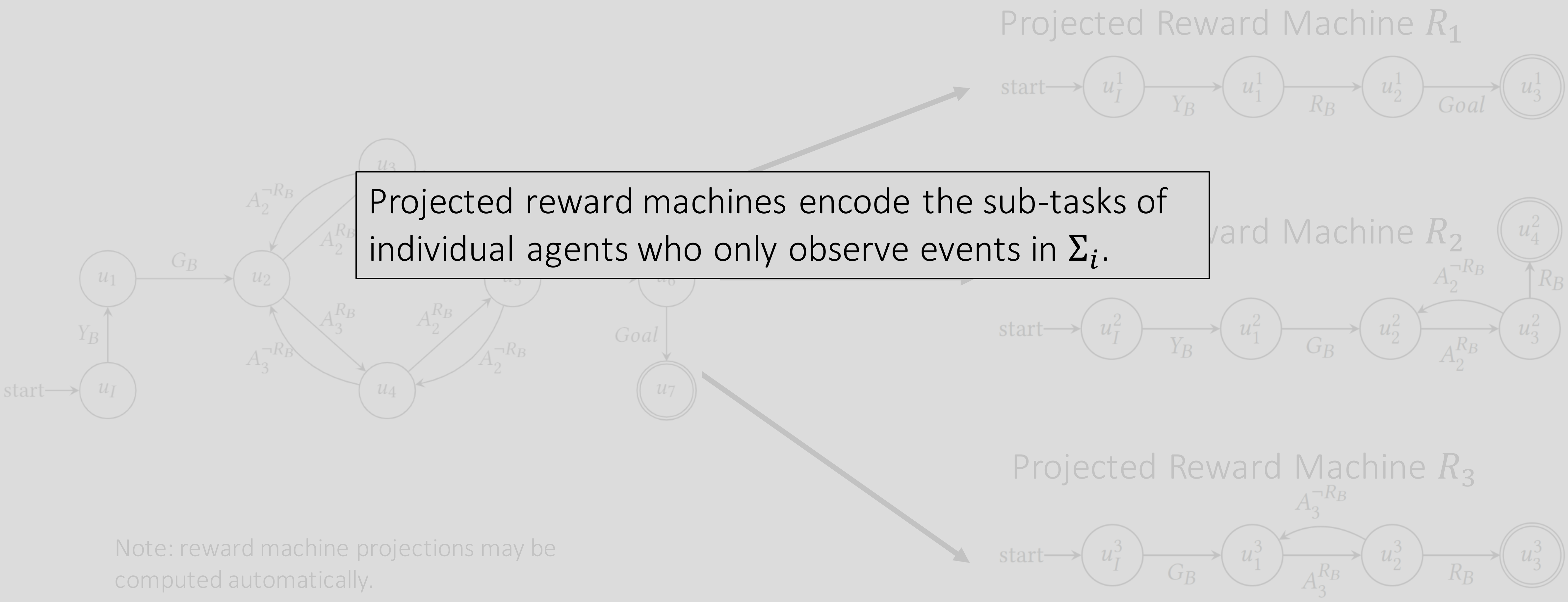
Projected Reward Machine R_3



Note: reward machine projections may be computed automatically.

Reward Machine Projection

How does the reward machine change if one only has access to events from $\Sigma_i \subseteq \Sigma$?



Projected reward machines encode the sub-tasks of individual agents who only observe events in Σ_i.

Note: reward machine projections may be computed automatically.

Problem Equivalence

When is the task described by the team reward machine equivalent to the composition of its projections?

Team Reward Machine R

Projected Reward Machine R_1

Projected Reward Machine R_2

...

Projected Reward Machine R_N

Problem Equivalence

When is the task described by the team reward machine equivalent to the composition of its projections?

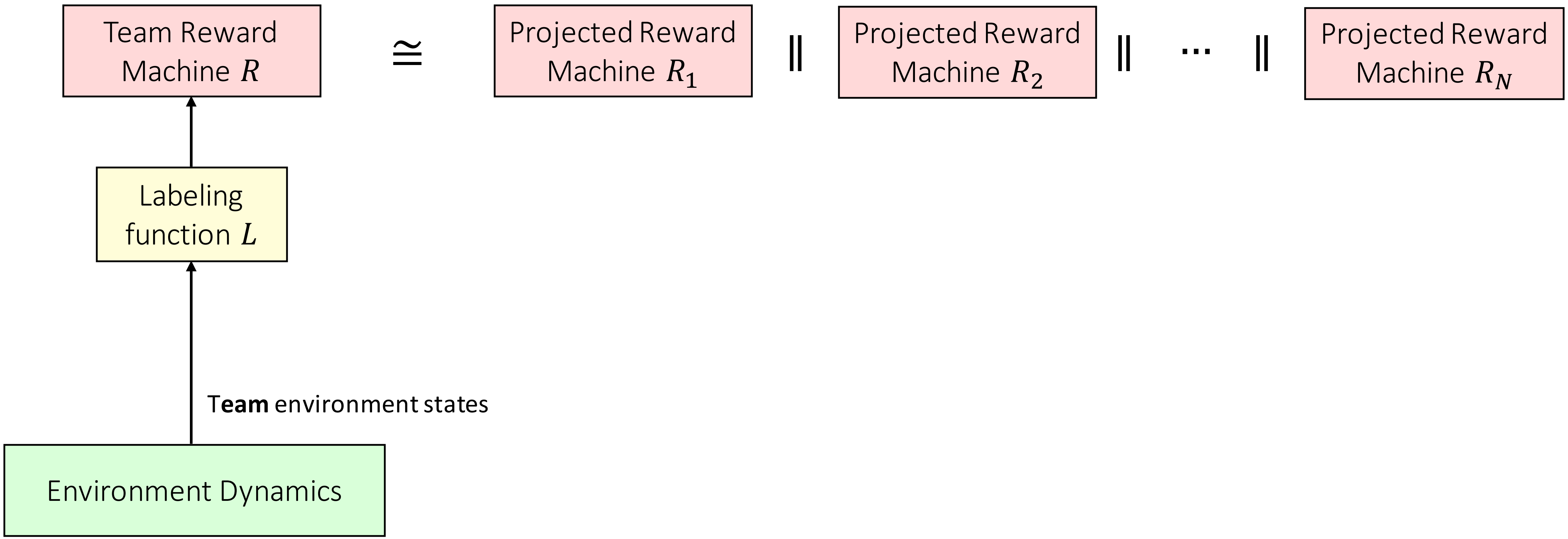
Team Reward Machine R $\cong$ Projected Reward Machine R_1 $\parallel$ Projected Reward Machine R_2 $\parallel$ $\cdots$ $\parallel$ Projected Reward Machine R_N

Bisimulation: Behavioral equivalence of reward machines

Parallel composition: Concurrent combination of reward machines

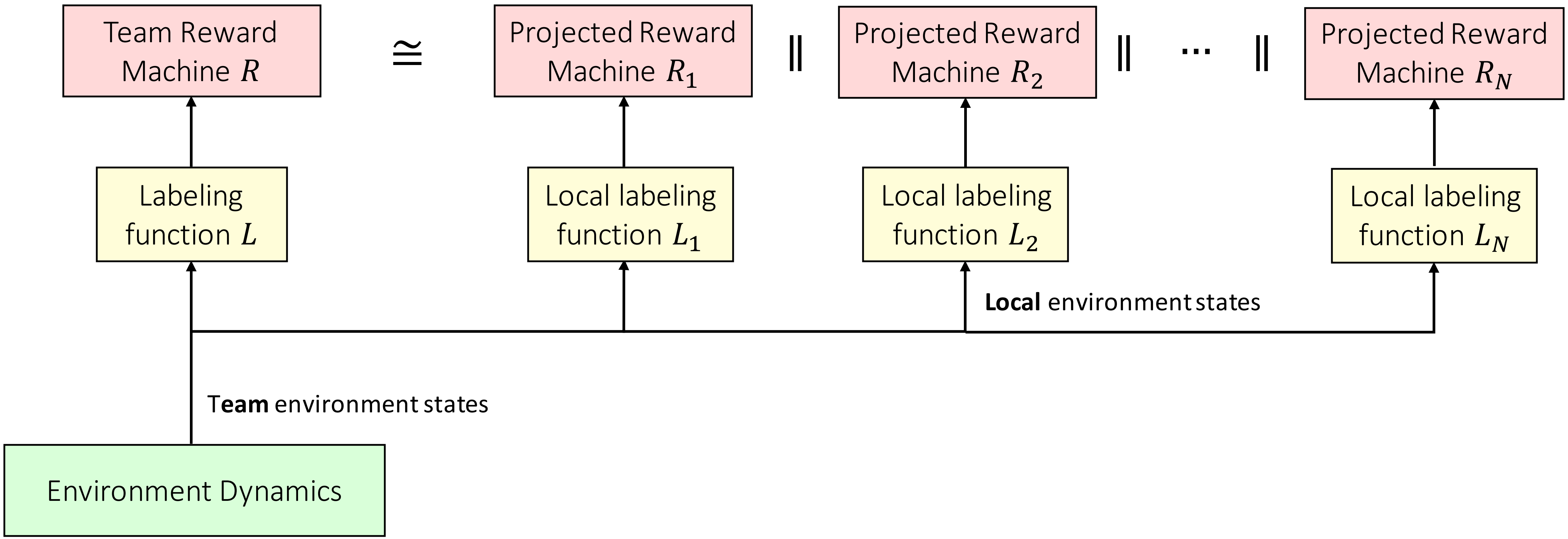
Local Labeling Functions

Connecting environment dynamics with projected reward machines?



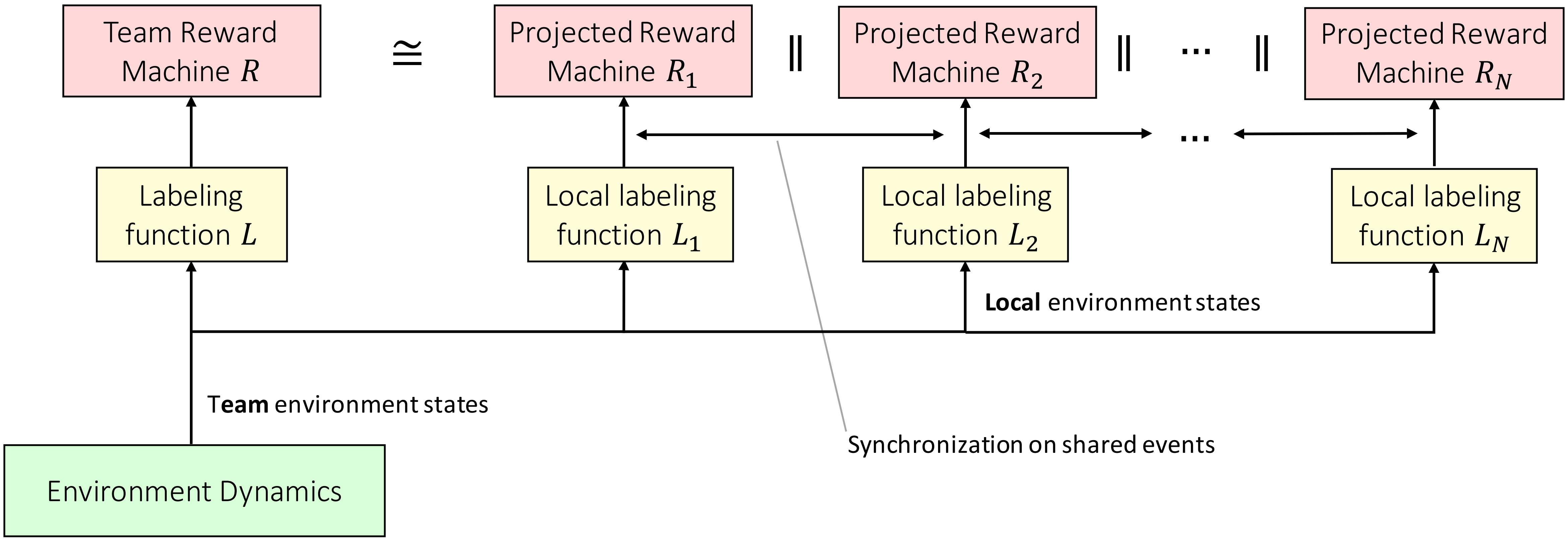
Local Labeling Functions

Connecting environment dynamics with projected reward machines?

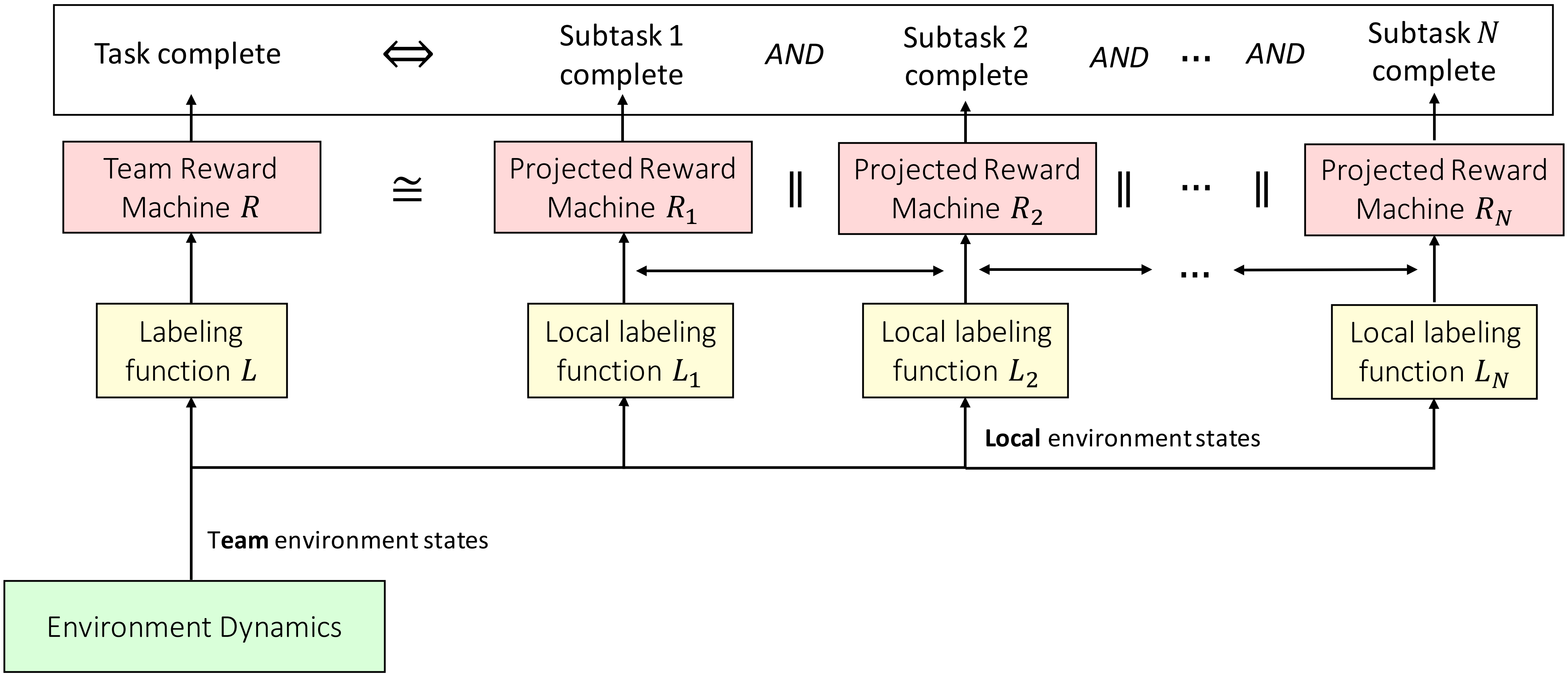


Local Labeling Functions

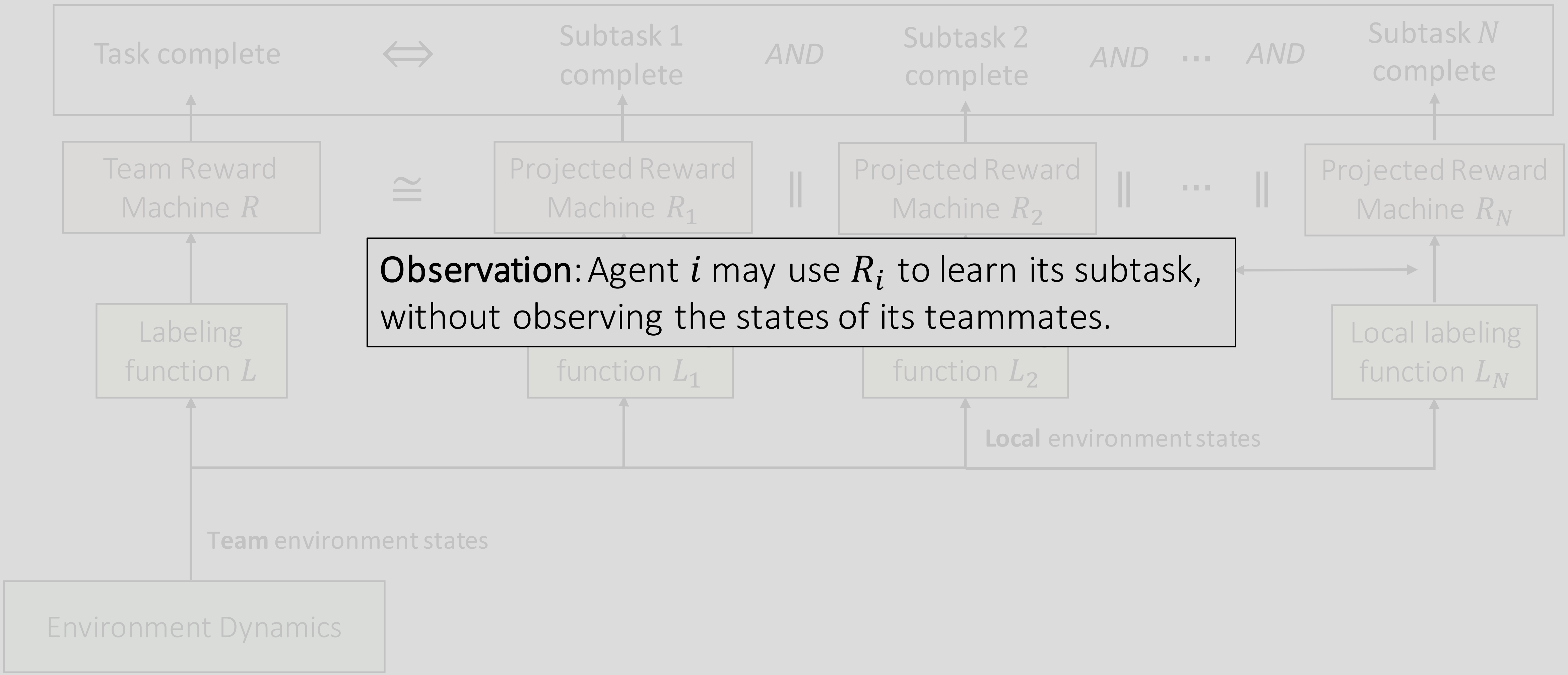
Connecting environment dynamics with projected reward machines?



Problem Equivalence



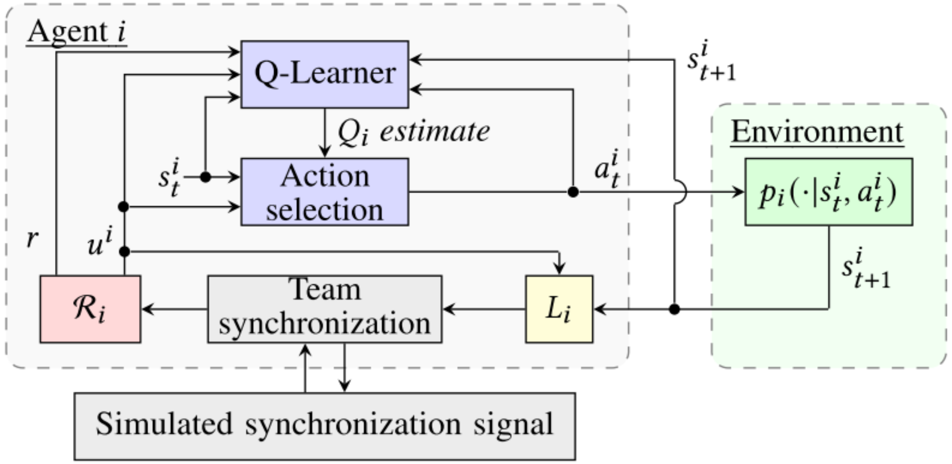
Problem Equivalence



Observation: Agent i may use R_i to learn its subtask, without observing the states of its teammates.

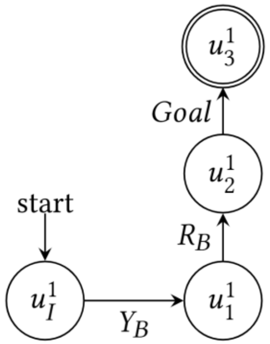
Decentralized Q-Learning with Projected Reward Machines (DQPRM)

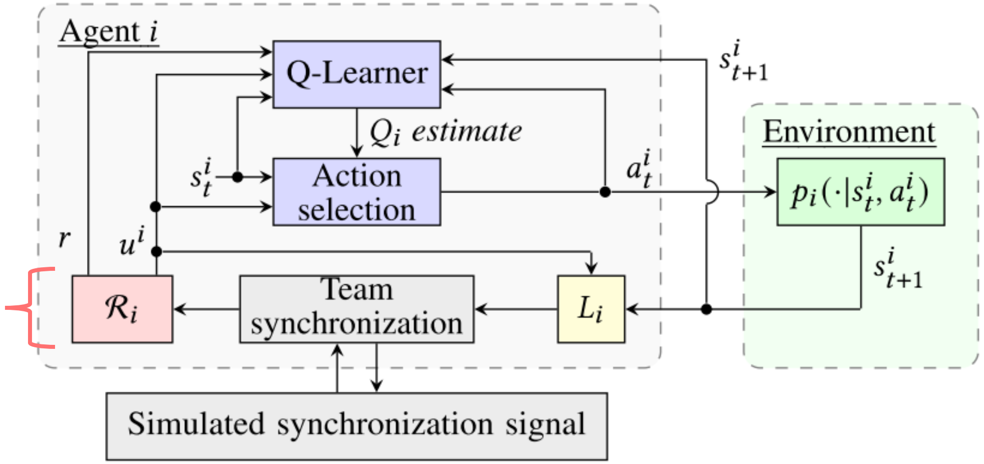
Agents learn to accomplish their subtasks **in the absence of teammates.**



Decentralized Q-Learning with Projected Reward Machines (DQPRM)

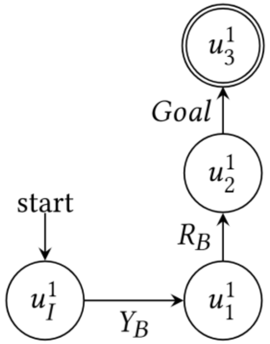
Agents learn to accomplish their subtasks **in the absence of teammates.**

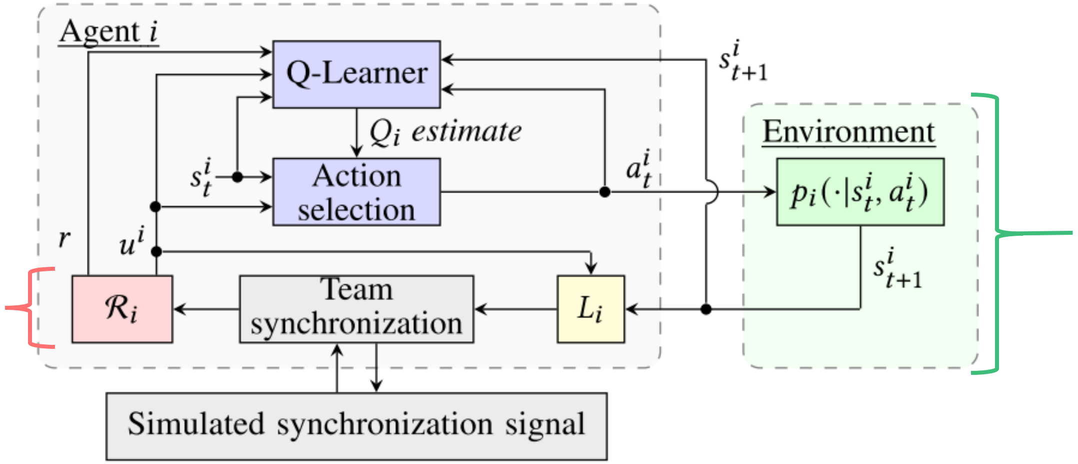


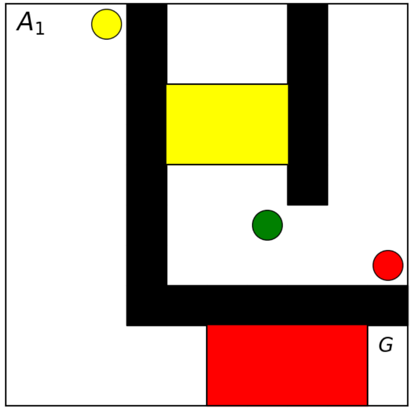


Decentralized Q-Learning with Projected Reward Machines (DQPRM)

Agents learn to accomplish their subtasks **in the absence of teammates.**

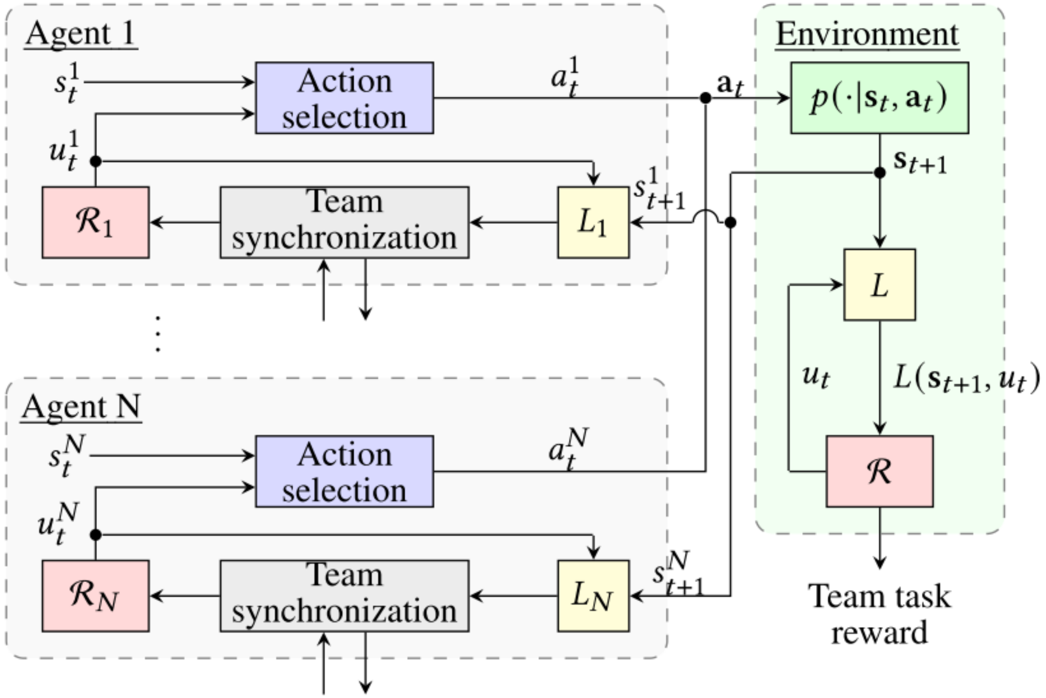






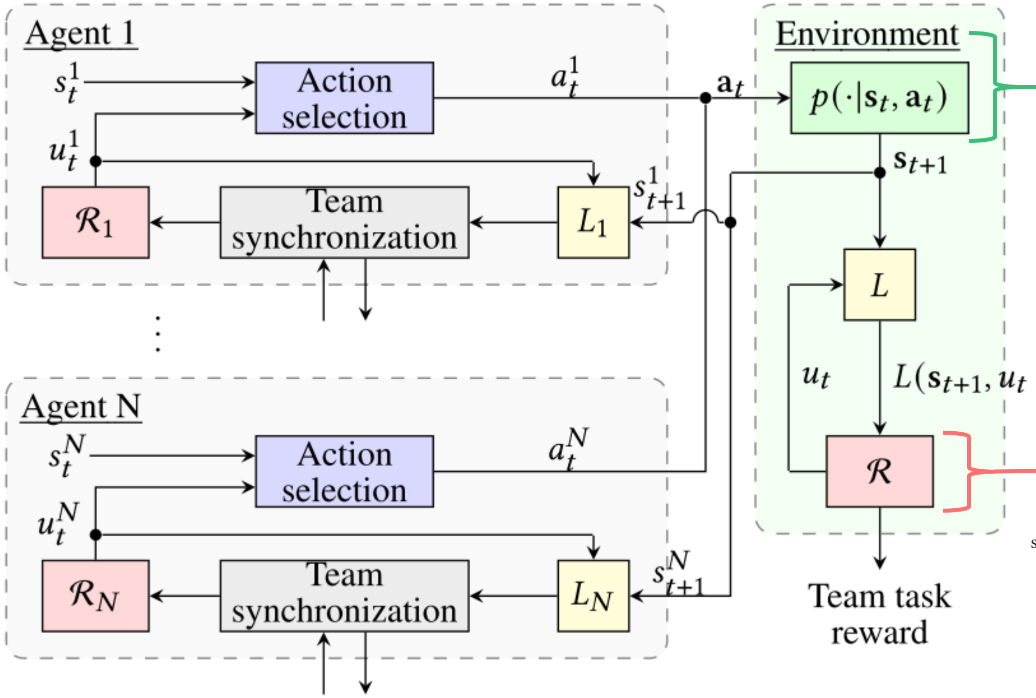
Decentralized Q-Learning with Projected Reward Machines (DQPRM)

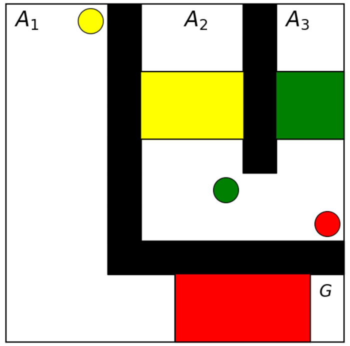
While ensuring the resulting composite behavior **accomplishes the team task.**

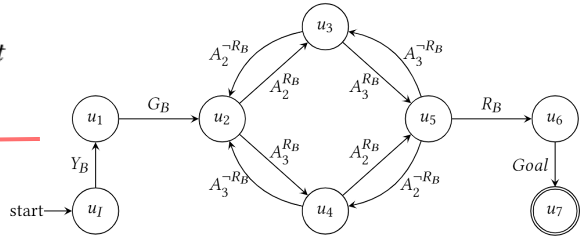


Decentralized Q-Learning with Projected Reward Machines (DQPRM)

While ensuring the resulting composite behavior **accomplishes the team task.**



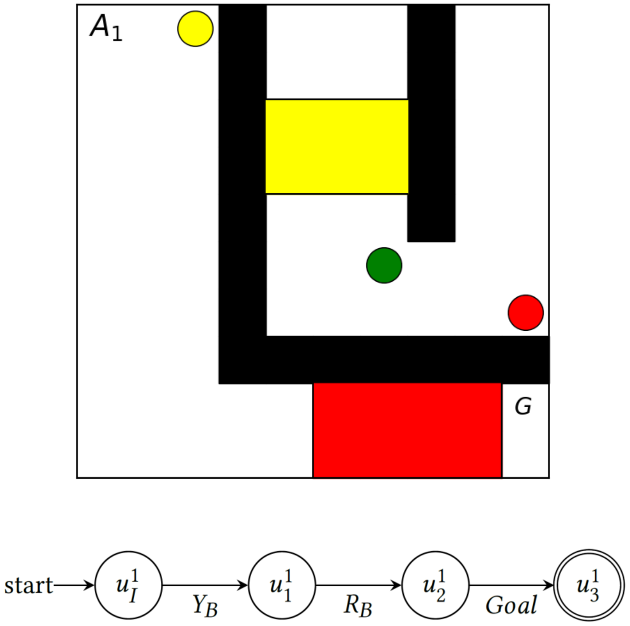




Decentralized Q-Learning with Projected Reward Machines (DQPRM)

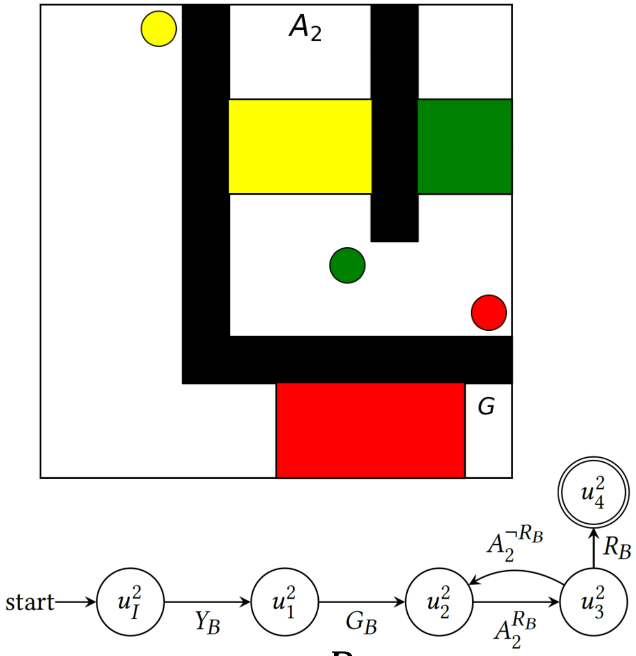
Agents learn to accomplish their subtasks **in the absence of teammates.**

Agent 1



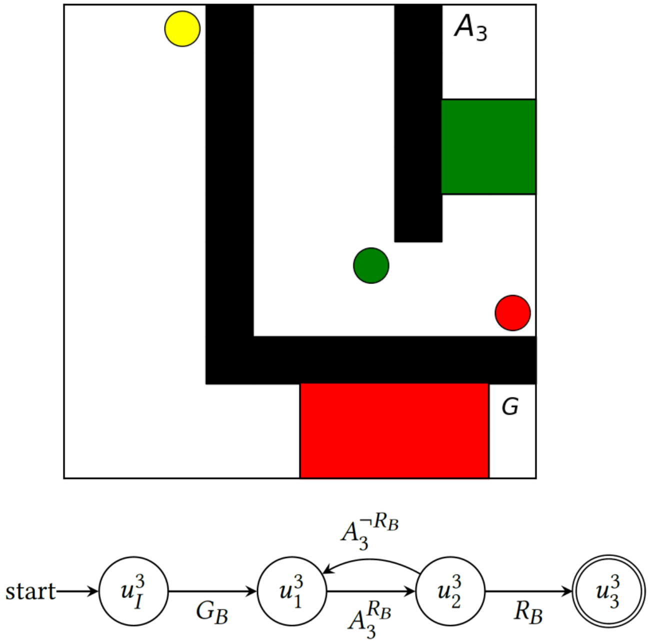
R_1

Agent 2



R_2

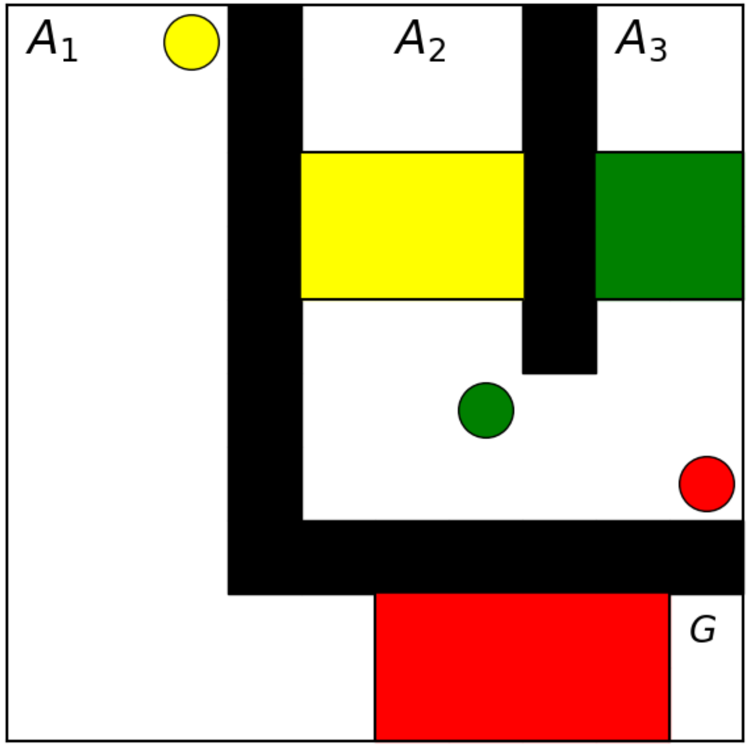
Agent 3

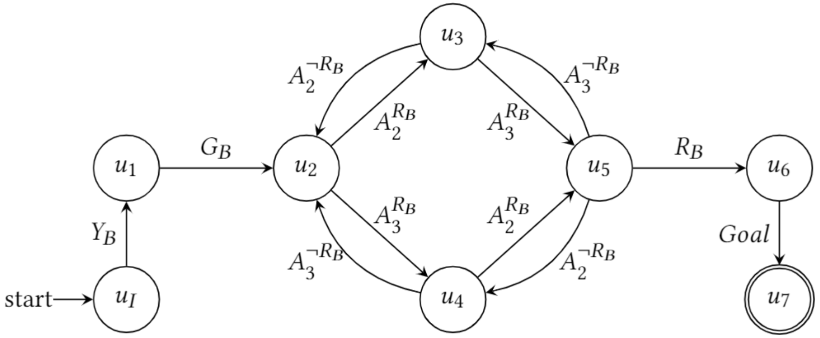


R_3

Decentralized Q-Learning with Projected Reward Machines (DQPRM)

While ensuring the resulting composite behavior **accomplishes the team task.**

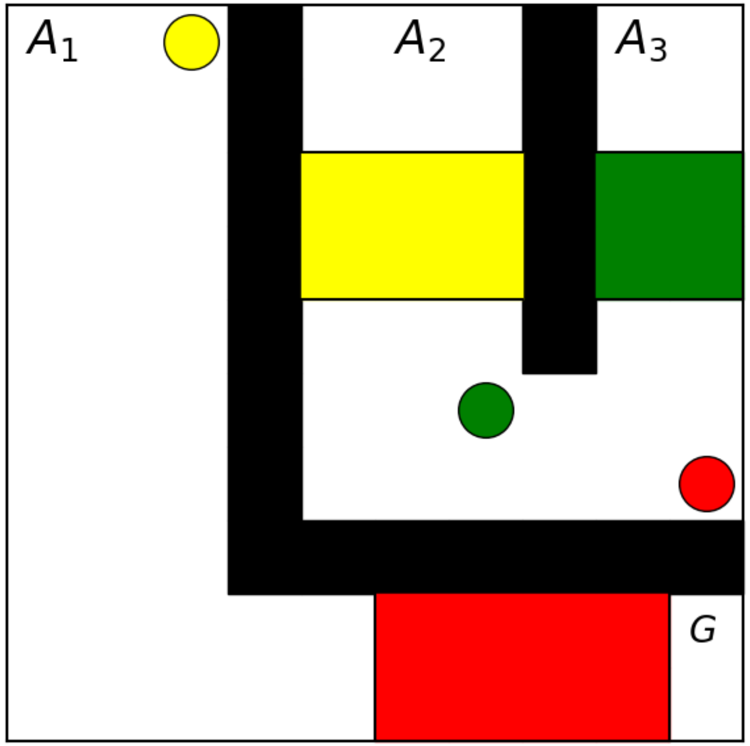


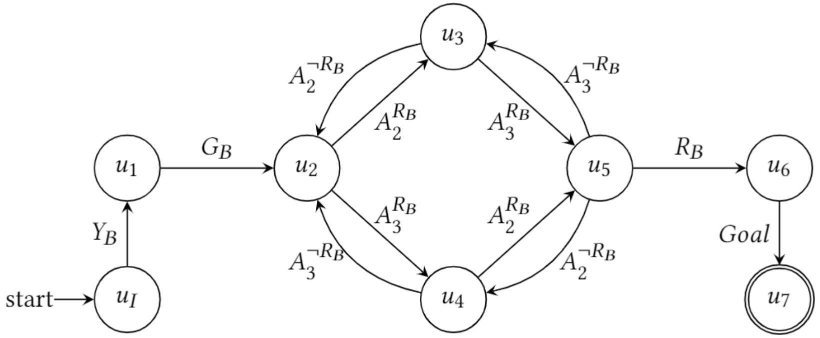


Team reward machine R

Decentralized Q-Learning with Projected Reward Machines (DQPRM)

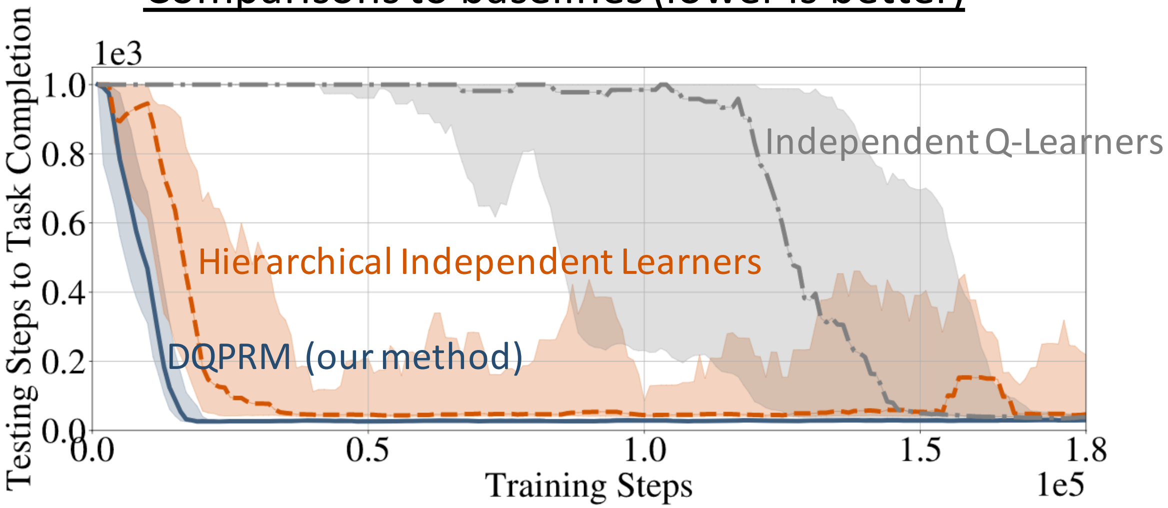
While ensuring the resulting composite behavior **accomplishes the team task.**





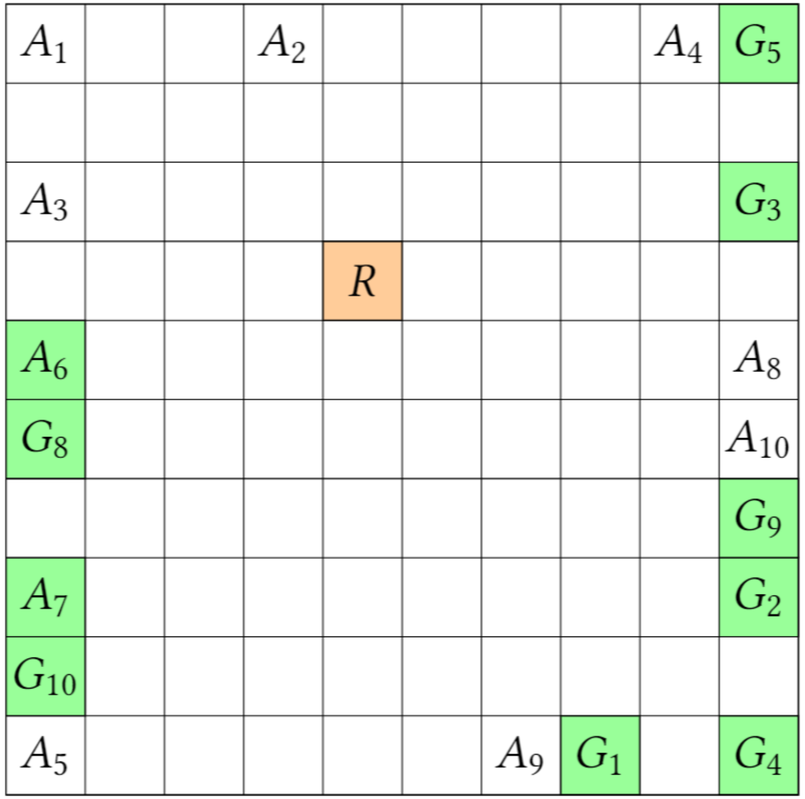
Team reward machine R

Comparisons to baselines (lower is better)



DQPRM Scaling with the Number of Agents

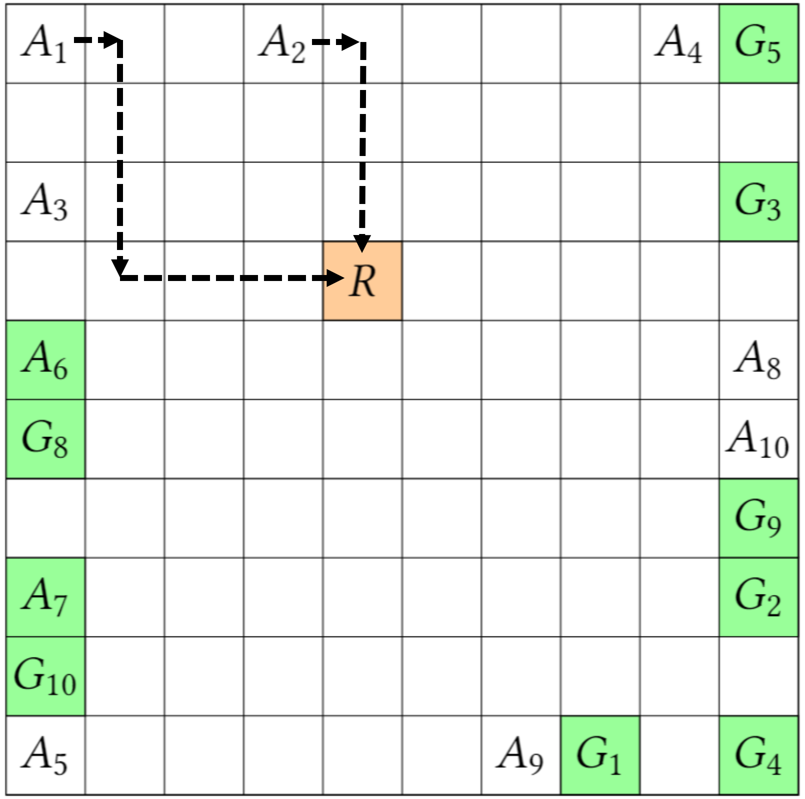
Rendezvous Experiment



Team Task: Each agent must *simultaneously* occupy the rendezvous location R, before proceeding to their respective goals G_i.

DQPRM Scaling with the Number of Agents

Rendezvous Experiment

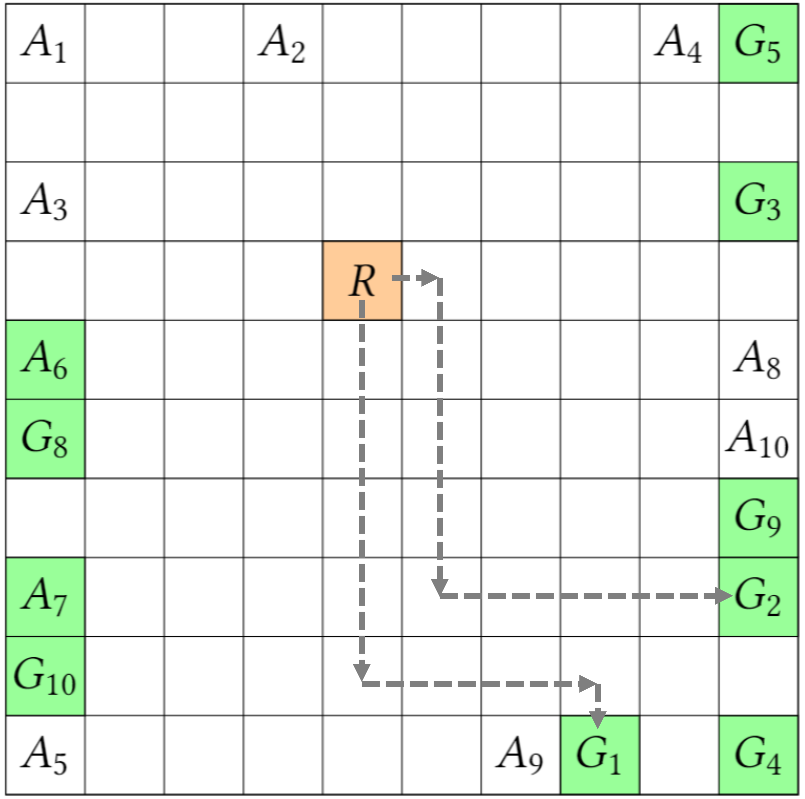


Team Task: Each agent must *simultaneously* occupy the rendezvous location R, before proceeding to their respective goals G_i.

- Two agent rendezvous.
- Ten agent rendezvous.

DQPRM Scaling with the Number of Agents

Rendezvous Experiment

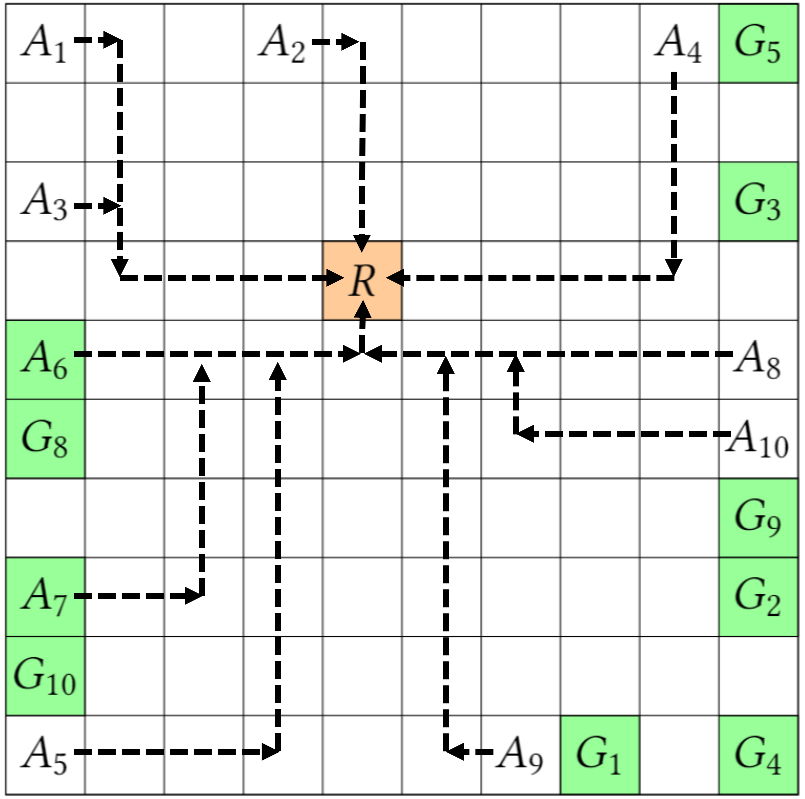


Team Task: Each agent must *simultaneously* occupy the rendezvous location R, before proceeding to their respective goals G_i.

- Two agent rendezvous.
- Ten agent rendezvous.

DQPRM Scaling with the Number of Agents

Rendezvous Experiment

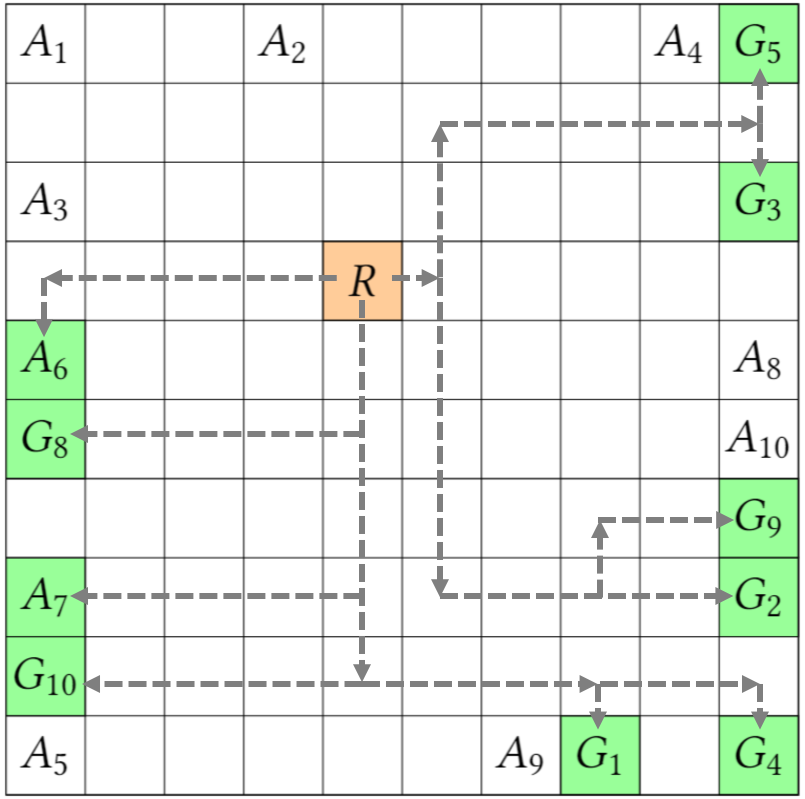


Team Task: Each agent must *simultaneously* occupy the rendezvous location R, before proceeding to their respective goals G_i.

- Two agent rendezvous.
- Ten agent rendezvous.

DQPRM Scaling with the Number of Agents

Rendezvous Experiment

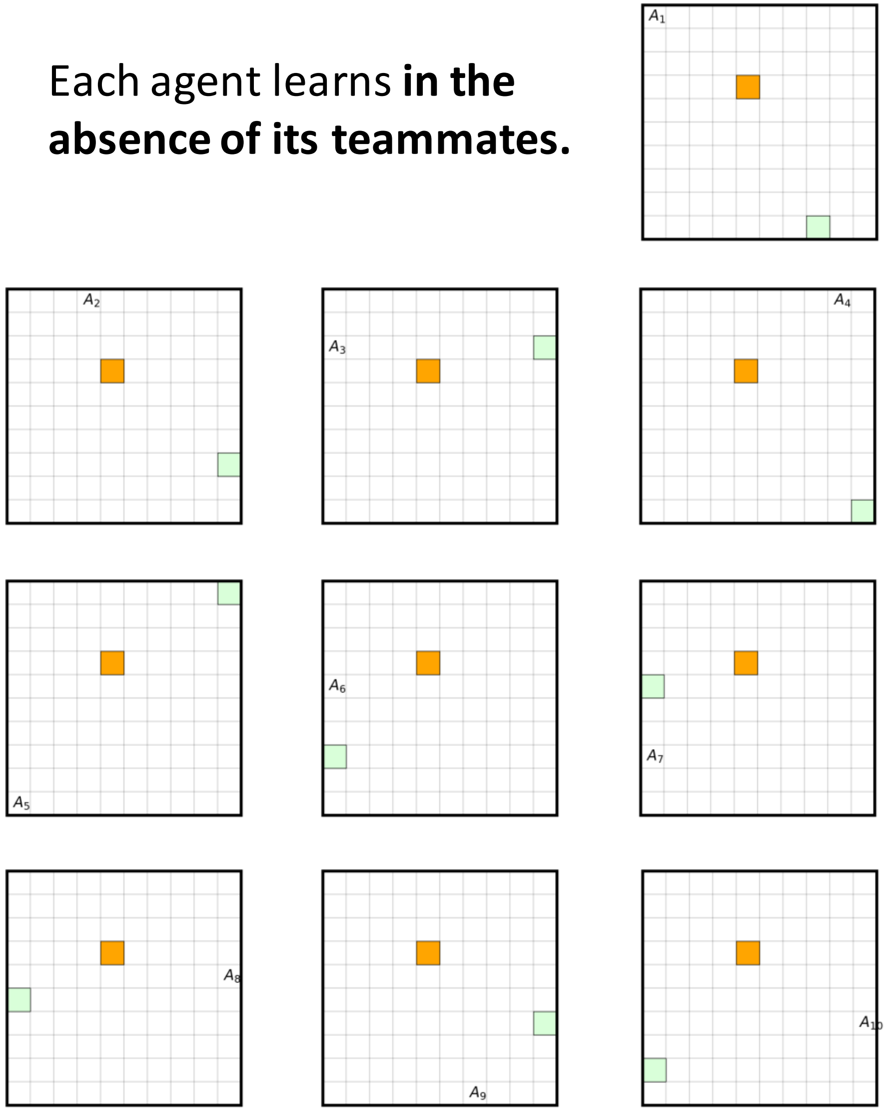


Team Task: Each agent must *simultaneously* occupy the rendezvous location R, before proceeding to their respective goals G_i.

- Two agent rendezvous.
- Ten agent rendezvous.

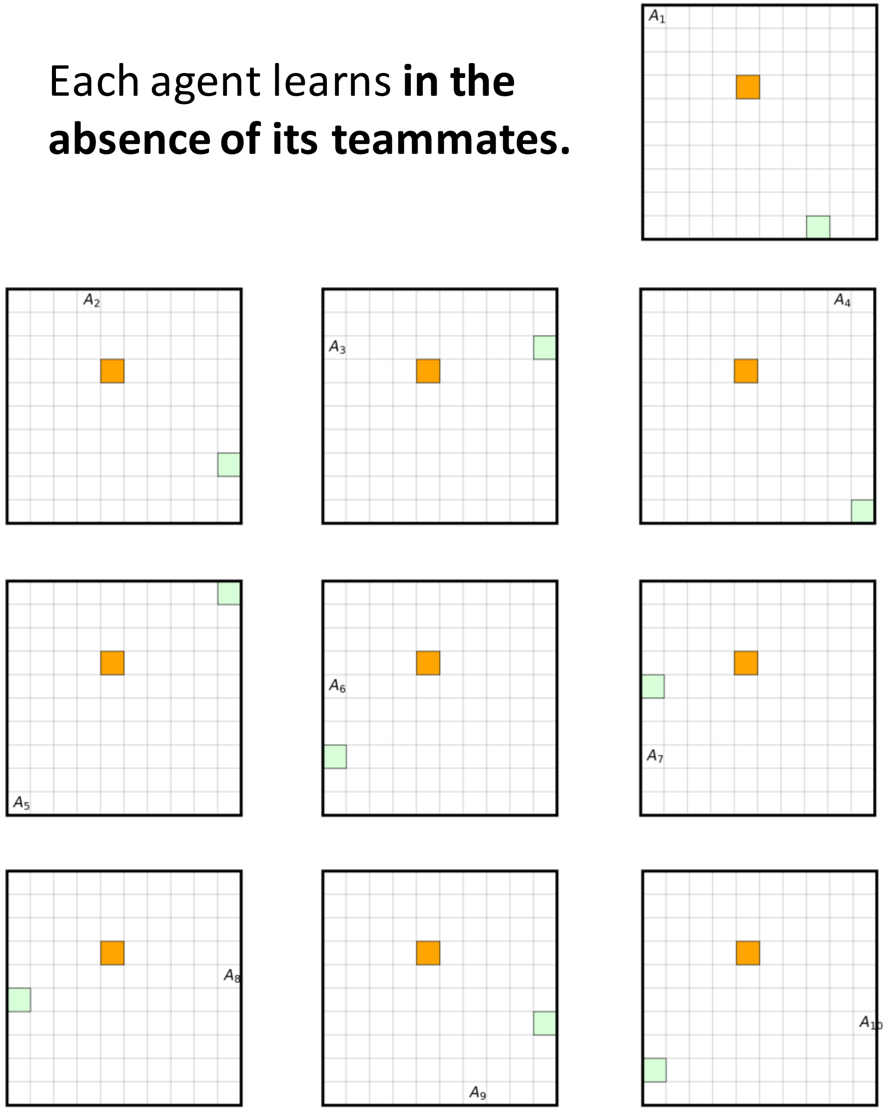
DQPRM scales well with the number of agents.

Each agent learns **in the absence of its teammates.**

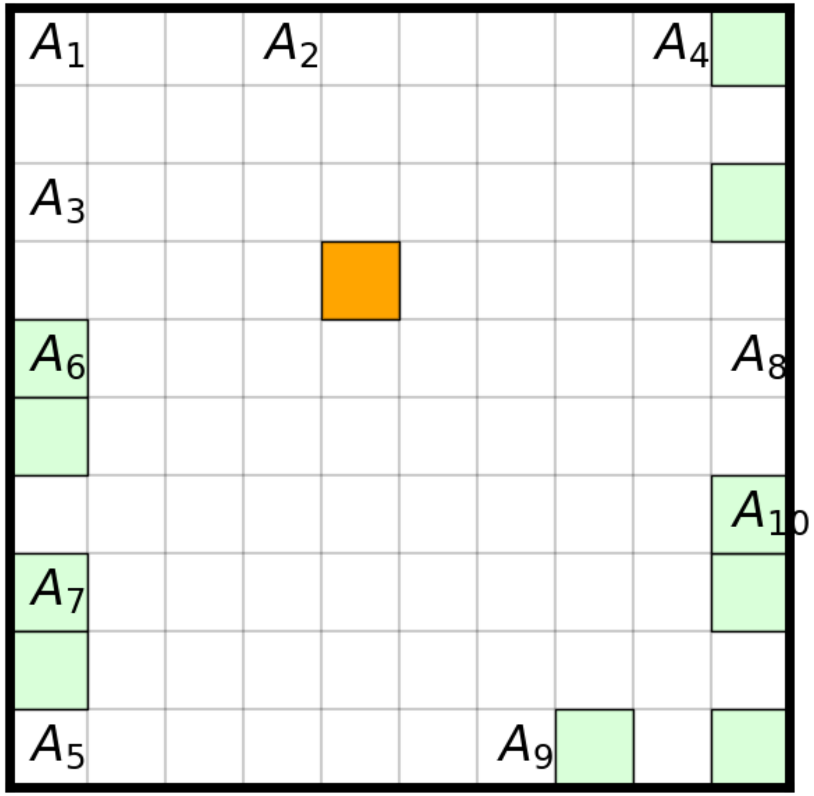


DQPRM scales well with the number of agents.

Each agent learns **in the absence of its teammates.**

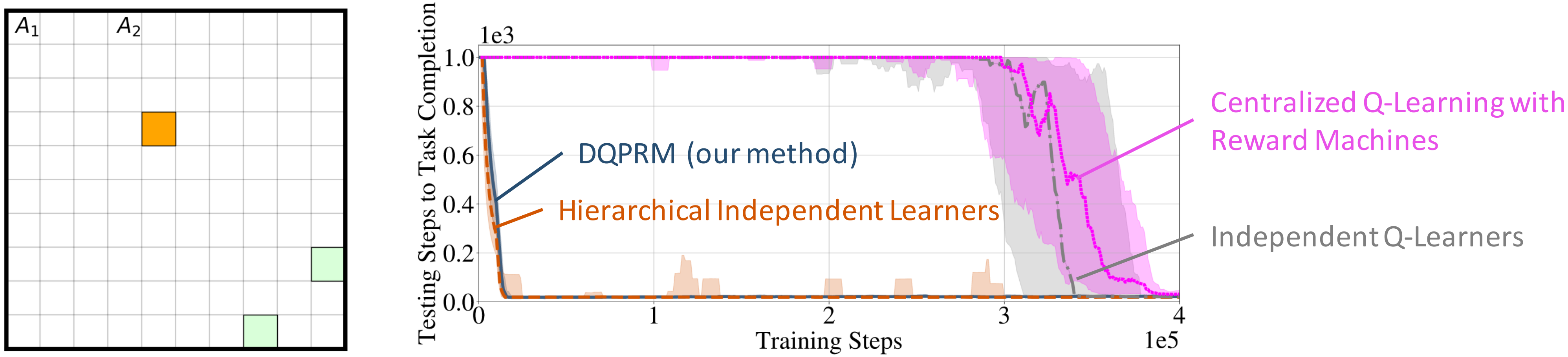


Composite behavior **accomplishes the team task.**

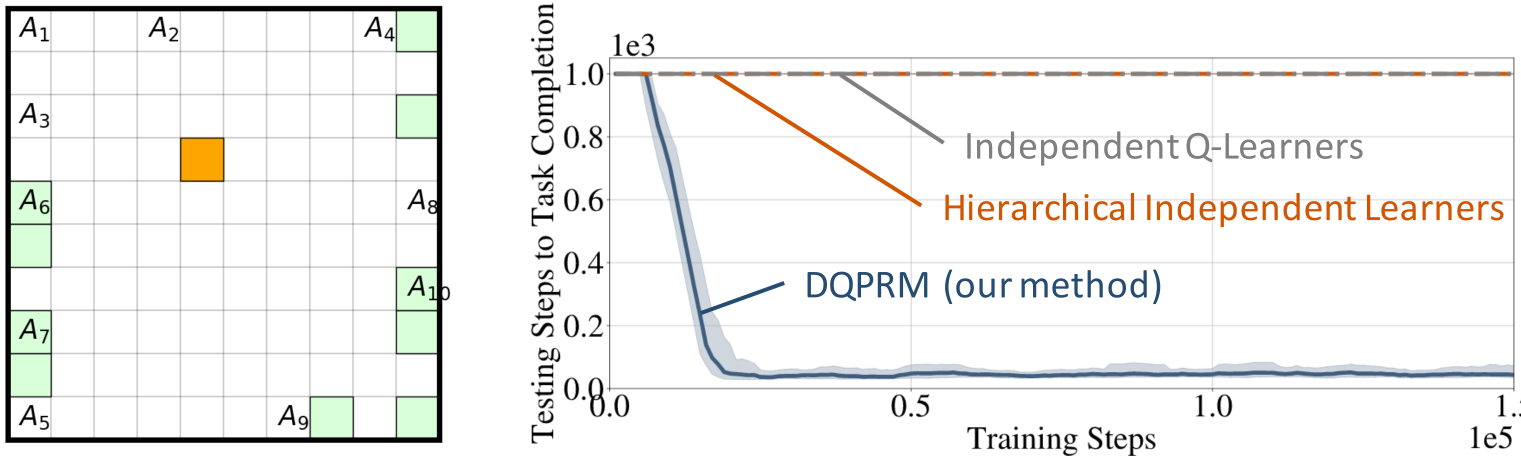


DQPRM scales well with the number of agents.

Two agent rendezvous



Ten agent rendezvous



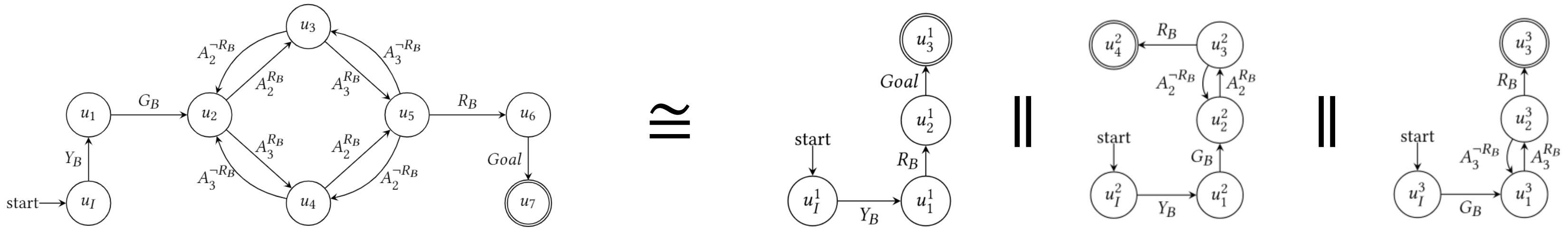
Reward machines for MARL task decomposition

Reward machines to **specify** and **decompose** cooperative RL tasks.

Reward machines for MARL task decomposition

Reward machines to **specify** and **decompose** cooperative RL tasks.

Conditions guaranteeing equivalence between original and decomposed tasks.

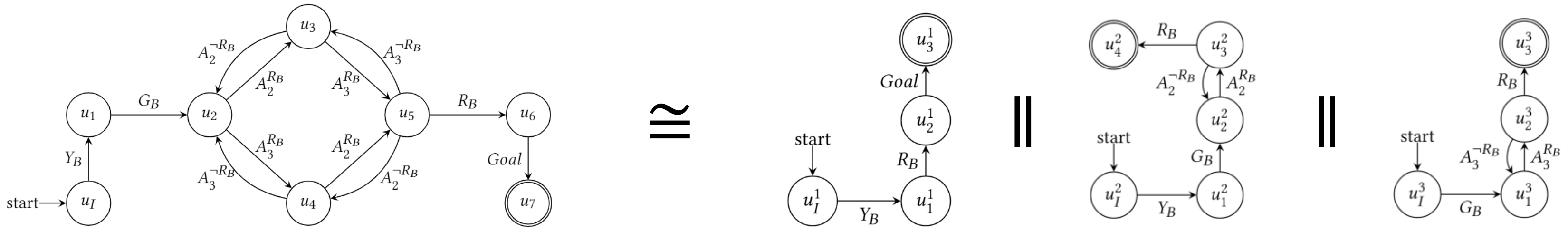


Cyrus Neary
cneary@utexas.edu

Reward machines for MARL task decomposition

Reward machines to **specify** and **decompose** cooperative RL tasks.

Conditions guaranteeing equivalence between original and decomposed tasks.



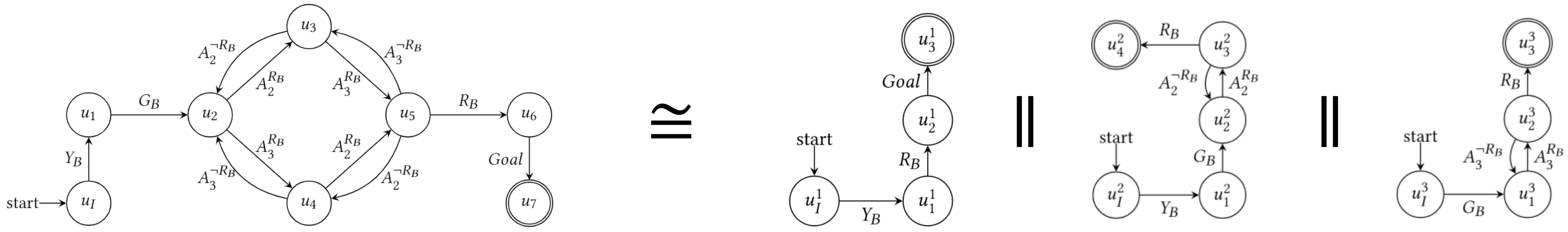
Decentralized Q-learning algorithm that trains each agent separately.

Cyrus Neary
cneary@utexas.edu

Reward machines for MARL task decomposition

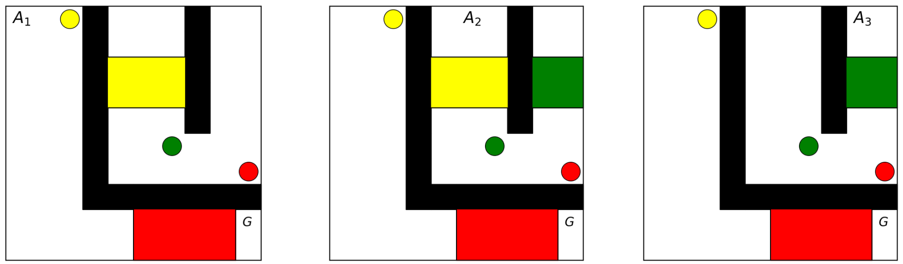
Reward machines to **specify** and **decompose** cooperative RL tasks.

Conditions guaranteeing equivalence between original and decomposed tasks.



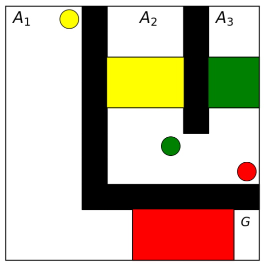
Decentralized Q-learning algorithm that trains each agent separately.

Training



Agent 1 Agent 2 Agent 3

Execution



Team

Cyrus Neary
cneary@utexas.edu