A APPENDIX

In this appendix, we will present additional experimental results; the design details of the SRMs and the symbolic constraints used in the experiments; a detailed experimental setup including the hyperparameters.

A.1 ADDITIONAL RESULTS

We show some addition experimental results in this section to answer the following questions.

E. Can arbitrarily concretized SRM effectively train RL agents?

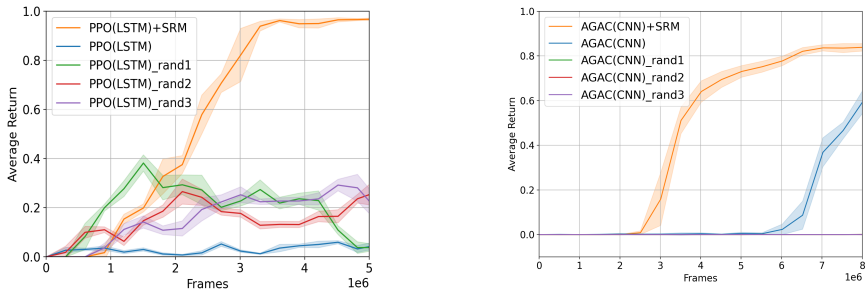
F. How much do the performance of Algorithm 1 depend on the designs of the SRMs?

For question **E**, we randomly generate hole assignments that satisfy the symbolic constraints for the SRMs of the DoorKey and KeyCorridor tasks. The SRMs are shown in Fig.8 and 9. The symbolic constraints contain the relational predicates as shown in Table.1 and 2. Those SRMs and symbolic constraints produce the main results in the main text. Now the assignments are generated by only optimizing the supervised objective J_{con} mentioned in the main text. The concretized SRMs are used for training RL policies in the following large DoorKey and KeyCorridor environments.

- **DoorKey-16x16** . In Fig.5a, we test three 3 randomly generated hole assignments for the SRM, each annotated by PPO(LSTM)_rand# . The PPO(LSTM) agents trained with those SRMs achieve certain level of performance than that trained with the default reward. However, the SRM concretized with a learned hole assignment, annotated by PPO(LSTM)+SRM, enables the agent to attain much higher performance with much lower amount of frames.
- **KeyCorridorS4R4** . we test 3 randomly generated hole assignments for the SRMs, each annotated by AGAC(CNN)_rand#. As in Fig.5b, the agents trained with the SRMs with random assignments do not perform at all. In contrast, the agent trained with the SRM that is concretized with a learned hole assignment achieves high performance with comparable amount of frames to that trained with the default reward.

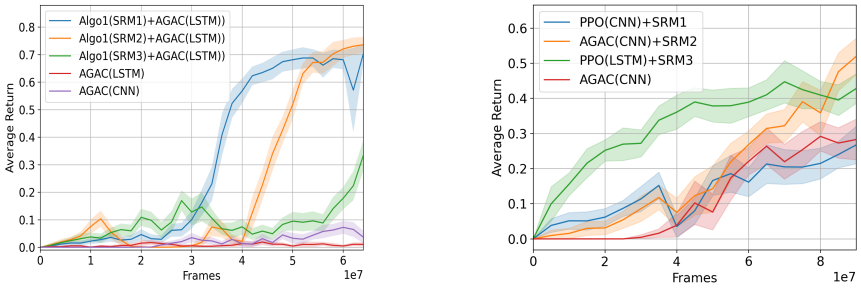
For question **F**, as mentioned in the main text we design three SRMs for the ObstructedMaze task. We will describe the difference between these SRMs in the next section. We run Algorithm 1 with those SRMs in the ObstructedMaze-2Dhlb environment and compare the results in Fig.6. In Fig.7b, we use those concretized SRMs to train RL agents in ObstructedMaze-Full. However, the SRM1 that achieves highest performance in Fig.7b is outperformed by two others.

Besides answering those two questions, we recall that we run Algorithm 1 in DoorKey and KeyCorridor tasks without symbolic constraint and with weaker symbolic constraint in the ablation study



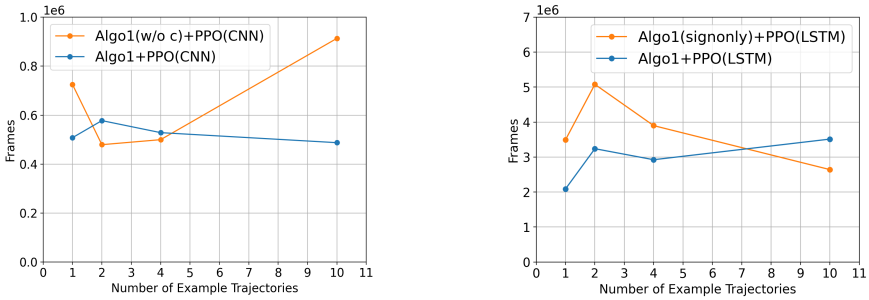
(a) DoorKey-16x16 (b) KeyCorridorS4R3

Figure 5: AGAC/PPO+SRM indicates that the hole assignments are learned via Algorithm 1; AGAC/PPO_rand# with an index # indicates that the holes are randomly assigned with some values that satisfy the symbolic constraint for that task. CNN and LSTM indicate the versions of the actor-critic networks.



(a) ObstructedMaze-2Dhlb (b) ObstructedMaze-Full

Figure 6: Algo1(SRM#)+AGAC(LSTM) with an index $\# = 1 \sim 3$ indicates running Algorithm 1 with those three designed SRMs and by using AGAC in line 4 of Algorithm 1. PPO/AGAC+SRM# indicates training RL agents with SRM# by using PPO or AGAC algorithm. CNN and LSTM indicate the versions of the actor-critic networks.



(a) DoorKey-8x8 (b) KeyCorridorS4R3

Figure 7: Algo1+PPO(CNN) indicates using PPO as the policy learning algorithm in line 4 of Algorithm 1; Algo1(w/o c)+PPO(CNN) indicates that running Algorithm 1 without symbolic constraint while using PPO(CNN) in line 4; Algo1(signonly)+PPO(CNN) indicates that running Algorithm 1 without symbolic constraint while using PPO(CNN) in line 4;CNN indicates CNN version of the actor-critic networks.

of the main text. Under the same conditions, we vary the number of demonstrations and check the number of frames needed for π_A to attain high performance. In Fig.7a and Fig.7b, we show that when the number of examples is reduced from 10 to 1, number of frames that Algorithm 1 needs to produce a policy with average return of at least 0.8 are not severely influenced.

A.2 Design Details of the SRMs

In this section, we show the diagrams of the SRMs as well as the symbolic constraints designed for the tasks. We will explain the design patterns in those SRMs in detail.

A.2.1 DoorKey Task

For readers convenience, we show the diagram of the SRM for DooKey in Fig.8. This SRM implicitly identifies an unlocking-door sub-task with two internal states "`Before Unlocking`" and "`After Unlocking`". The transitions are designed mostly based on high level human insights represented in first order logic: a) $(\texttt{Reach\_Goal@t}) \mapsto \exists \texttt{t}_1 < \texttt{t}. \exists \texttt{t}_2 < \texttt{t}_1. (\texttt{Unlock\_Door@t}_1) \wedge (\texttt{Pick\_up\_Key@t}_2)$ where `@t` indicates that the predicate preceding it, e.g., `Reach_Goal`, operates on the time step t of the trajectory τ; b) $\forall \texttt{t} \in [\texttt{t}_1, \texttt{t}_2]. (\texttt{Drop\_Key@t}_1, \texttt{Before Unlocking}) \wedge (\neg \texttt{Pick\_up\_Key@t}) \wedge (\texttt{Pick\_up\_Key@t}_2) \mapsto \forall \texttt{t}' \in [\texttt{t}_1, \texttt{t}_2]. (\neg \texttt{Unlock\_Door@t}')$ where we additionally integrate the internal state, i.e., "`Before Unlocking`", next to $\texttt{@t}_1$, to indicate the internal

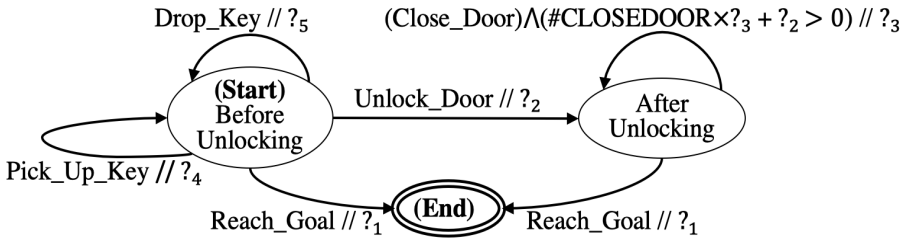


Figure 8: The diagram of the SRM designed for the DoorKey task.

Properties	**Predicates**
$[\mu_1]$Reward reaching the goal	$\bigwedge_{\texttt{id}=1}^{5} (?_{\texttt{id}} \leq ?_1)$
$[\mu_2]$Penalize dropping unused key	$?_5 + ?_4 \leq 0$
$[\mu_3]$Reward unlocking door	$\bigwedge_{\texttt{id}=2}^{5} (?_{\texttt{id}} \leq ?_2)$
$[\mu_4]$Penalty for closing door	$?_3 \leq 0$
$[\mu_5]$Mildly penalize door toggling	$?_3 + ?_2 \leq 0$

Table 1: The correspondence between properties and atomic predicates for the DoorKey SRM in Fig.1b

state at the time step t_1. The predicate $\#\texttt{CLOSEDOOR} \times ?_3 + ?_2 > 0$ in Fig.1b is introduced with due consideration of avoiding overly penalizing the agent for closing the door, which behavior is redundant for the task. The underlying idea is: *if the reward function penalized an under-trained RL agent for every door closing behavior with some high penalty* $?_3 < 0$ *for a total of* $\#\texttt{CLOSEDOOR}$ *amount of times, and the accumulated penalty* $\#\texttt{CLOSEDOOR} \times ?_3$ *outweighed the reward* $?_2 > 0$ *for unlocking the door, then the agent in practice might be inclined to reside away from the door for good.* The SRM in Fig.1b simply upper-bounds the accumulated penalty to avoid negative effects in practice. Then we show the atomic predicates in the symbolic constraint for this task in Table.1. The final symbolic constraint is $c = \bigwedge_{i=1}^{5} \mu_i$. We omit the explanation for the symbolic constraint since the atomic predicates are self-explanatory.

A.2.2 KeyCorridor Task

We depict in Fig.9 the diagram of the SRM designed for this task. Due to the added complexity in this task in comparison with the DoorKey task, two sub-tasks, finding-key and unlocking-door, are implicitly established by using three internal states "`Before Finding Key`", "`Before Unlocking`" and "`After_Unlocking`". Some important first order logic formulas that hold in most situations in the KeyCorridor task include: a) $(\texttt{Pick\_Up\_Target@t}_1) \wedge (\texttt{Unlock\_Door@t}_2) \mapsto \exists \texttt{t} \in [\texttt{t}_2, \texttt{t}_1].(\texttt{Drop\_Key@t})$;

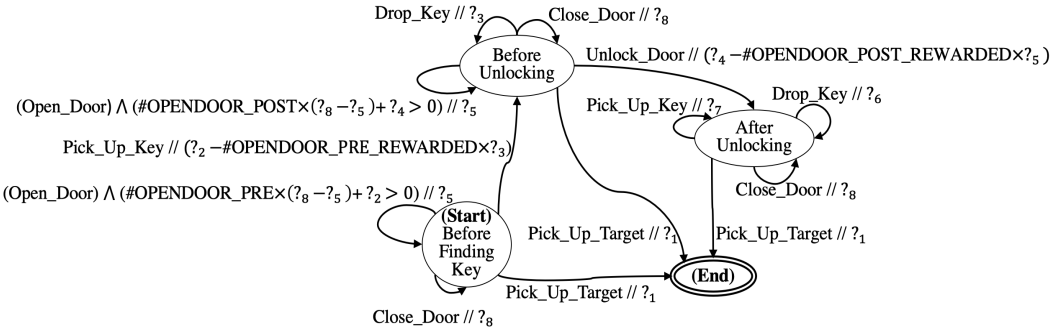


Figure 9: The diagram of the SRM designed for the KeyCorridor task.

Properties	**(Retaional) Predicates**	**(Non-Relational) Predicates**
$[\mu_1]$Reward picking up ball	$\bigwedge_{\texttt{id}=2}^{8}(?_{\texttt{id}} \leq ?_1)$	$?_1 \geq 0$
$[\mu_2]$Reward 1st time picking up key	$?_2 \geq 0$	$?_2 \geq 0$
$[\mu_3]$Reward dropping used key	$?_3 \geq 0$	$?_3 \geq 0$
$[\mu_4]$Reward unlocking door	$?_4 \geq 0$	$?_4 \geq 0$
$[\mu_5]$Encourage opening door	$?_5 \geq 0$	$?_5 \geq 0$
$[\mu_6]$Penalize meaningless move	$?_8 \leq 0$	$?_8 \leq 0$
$[\mu_7]$Moderately reward opening door	$?_5 - ?_8 \leq ?_2$	
$[\mu_8]$Penalize dropping unused key	$?_2 + ?_6 \leq 0$	$?_6 \leq 0$
$[\mu_9]$Penalize picking up used key	$?_3 + ?_7 \leq 0$	$?_7 \leq 0$

Table 2: The correspondence between properties and the relational and non-relational atomic predicates for the SRM of KeyCorridor in Fig.9

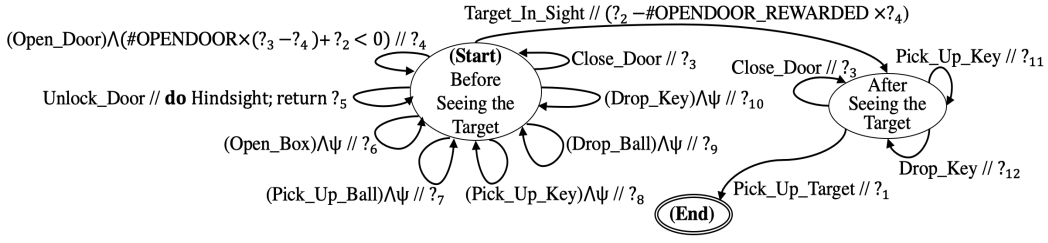
b) $\forall \texttt{t}' < \texttt{t}.(\texttt{Pick\_Up\_Key@t}) \wedge (\neg\texttt{Pick\_Up\_Key@t}') \mapsto \exists \texttt{t}'' < \texttt{t}.(\texttt{Open\_a\_Door@t}'')$; c) $(\texttt{Unlock\_Door@t}) \mapsto \exists \texttt{t}' < \texttt{t}.(\texttt{Open\_a\_Door@t}'')$. Regarding the implication a, two predicates `Pick_Up_Key@t` and `Drop_Key@t` are added at the internal state "`After Unlocking`" to govern the rewards returned for their respectively concerned behaviors after the door is unlocked. As for the implications b and c, the caveat is to determine the utility of each door opening behavior. A designer may go to one extremity by rewarding every door opening behavior with some constant, which, however, either represses exploration by penalizing opening door, or oppositely raises reward hacking, i.e., agent accumulates reward by exhaustively searching for doors to open. Alternatively, the designer may go to another extremity by carrying out a motion planning and specify the solution in the SRM, which, however, is cumbersome and cannot be generalized. In this paper, we highlight a economical design pattern to circumvent such non-determinism.

As shown in Fig.9, before the agent accomplishes the finding-key sub-task, i.e., in the "`Before Finding Key`" internal state, once the agent opens a door, the predicate $\#\texttt{OPENDOOR\_PRE} \times (?_8 - ?_5) + ?_2 > 0?$ checks whether the total reward gained from opening doors is about to exceed a threshold. The counter $\#\texttt{OPENDOOR\_PRE}$ counts the number of times that agent opens doors prior to the agent finding the key; the variable $?_8$ is expected to be a penalty for the agent closing a door, which is redundant. By introducing $?_8$, we specify that even if the agent closed doors instead of opening doors for equal number $\#\texttt{CLOSEDOOR\_PRE} \equiv \#\texttt{OPENDOOR\_PRE}$ of times, the agent could still gain positive net reward by finishing the finding-key sub-task, i.e., $\#\texttt{CLOSEDOOR\_PRE} \times ?_8 + ?_2 \geq \#\texttt{OPENDOOR\_PRE} \times ?_5$. When the agent accomplishes the finding-key sub-task, i.e., transitioning to the "`Before Unlocking`" internal state, the reward $?_2 - \#\texttt{OPENDOOR\_PRE\_REWARDED} \times ?_3$ subtracts the reward hitherto gained from opening doors with $\#\texttt{OPENDOOR\_PRE\_REWARDED} \leq \#\texttt{OPENDOOR\_PRE}$ counting the number of times that door opening behaviors are indeed awarded prior to the agent finding the key. In some sense, this approach amortizes the reward $?_2$ for finishing the finding-key sub-task over the door opening behaviors. The amortized reward $\#\texttt{OPENDOOR\_PRE\_REWARDED} \times ?_3$ cannot exceed $?_2$ and should be deducted from $?_2$. The same idea is adopted to award the door opening behaviors prior to the agent unlocking the door. The counter $\#\texttt{OPENDOOOR\_POST}$ in Fig.9 counts the number of times that agent opens doors after the agent finding the key prior to the agent unlocking the door; $\#\texttt{OPENDOOR\_PRE\_REWARDED}$ counts the number of times that door opening behaviors are awarded within that time interval. Apparently, such design pattern is convenient enough to be implemented via symbolic means. The challenge, however, remains to properly determine values for $?_{\texttt{id}}$'s. Then we show the atomic predicates in the symbolic constraint for this task in Table.2. The final symbolic constraint is $c = \bigwedge_{i=1}^{9} \mu_i$.

A.2.3 OBSTRUCTEDMAZE TASK

Fig.10 shows the diagram of the reward function designed for the ObstructedMaze task. Despite of the complexity of task, there are only three internal states, "**(Start)**", "`After Seeing the Target`" and "**(End)**". This is because only specifying the the sub-tasks is far from adequate for this task.

Once the "`After Seeing the Target`" state is reached, the reward function only concerns whether the agent drops or picks up a key or the target. In the "**(Start)**" state, the reward function views each door unlocking behavior as a milestone. If the agent does not unlock a



ψ: =?5 +#OPENBOX×(?3 −?6)+#PICKUPBALL ×(?3 −?7) + #PICKUPKEY×(?3 −?8)+#DROPBALL×(?3 −?9)+#DROPKEY×(?3 −?10) > 0

```
Hindsight :=  let t0 = Last_Time_Unlock_Door          in  Replay_Buffer.Rewards[t0:t]= 0  ;
              let t1 = Last_Time_Open_Right_Box       in  Replay_Buffer.Rewards[t1] = ?6  ;
              let t2 = Last_Time_Pick_Up_Right_Ball   in  Replay_Buffer.Rewards[t2] = ?7  ;
              let t3 = Last_Time_Pick_Up_Key          in  Replay_Buffer.Rewards[t3] = ?8  ;
              let t4 = Last_Time_Drop_Ball            in  Replay_Buffer.Rewards[t4] = ?9  ;
              let t5 = Last_Time_Drop_Wrong_Ball      in  Replay_Buffer.Rewards[t5] = ?10 ;
```

Figure 10: The diagram of the SRM designed for the ObstructedMaze task.

door at the present time step, the SRM awards the following agent behaviors: opening a box, picking up a ball, picking up a key, dropping a ball, dropping a key through a proposition $?_5 + \#\texttt{OPENBOX} \times (?_3 - ?_6) + \#\texttt{PICKUPBALL} \times (?_3 - ?_7) + \ldots > 0$ which bounds the number of times that the those behaviors are awarded. The counters $\#\texttt{OPENBOX}, \#\texttt{PICKUPBALL}\ldots$ only count the number of respectively concerned behaviors between two successive door unlocking behaviors by resetting themselves to 0 once the agent unlocks a door. Thus far the design pattern is still similar to that adopted in the KeyCorridor tasks. What makes a difference here is that we assume the SRM to have access to the replay buffer of the agent policy, annotated as `Replay_Buffer`. Suppose that in some time step `t` the agent unlocks a `C` colored door located at coordinate `X`, the SRM locates the last time step when the agent unlocked a door. Then it reassigns the rewards to 0 for all the opening a box, picking up a ball, picking up a key, dropping a ball, dropping a key behaviors stored in `Replay_Buffer` ever since that last door locking time step till the present time step. Then it identifies the time steps of four milestone behaviors based on the following human insights represented in first order logic: a) $(\texttt{Unlock\_C\_Colored\_Door@t}) \mapsto \exists \texttt{t}_1 < \texttt{t}.(\texttt{Open\_Box@t}_1) \wedge (\texttt{Find\_C\_Colored\_Key@t}_1 + 1)$, i.e., in time step $\texttt{t}_1$ the agent opened the box that contains the key for this C colored door; b) $(\texttt{Unlock\_X\_Located\_Door@t}) \mapsto \exists \texttt{t}_2 < \texttt{t}.\forall \texttt{t}_2' > \texttt{t}_2.(\texttt{Pick\_Up\_X\_Located\_Ball@t}_2) \wedge (\neg \texttt{Pick\_Up\_X\_Located\_Ball@t}_2')$, i.e., in time step $\texttt{t}_2$ the agent picked up the ball obstructing this door at position X for the last time; c) $(\texttt{Unlock\_C\_Colored\_Door@t}) \mapsto \exists \texttt{t}_3 < \texttt{t}.\forall \texttt{t}_3' \in [\texttt{t}_3, \texttt{t}].(\texttt{Pick\_Up\_C\_Colored\_Key@t}_3) \wedge (\neg \texttt{Pick\_Up\_C\_Colored\_Key@t}_3')$, i.e., in time step $\texttt{t}_3$ the agent picked up the key for this C colored door for the last time; d) $(\texttt{Unlock\_Door@t}) \mapsto \exists \texttt{t}_4 < \texttt{t}.\forall \texttt{t}_4' \in [\texttt{t}_4, \texttt{t}].(\texttt{Drop\_Ball@t}_4) \wedge (\neg \texttt{Drop\_Ball@t}_4')$, i.e., in time step $\texttt{t}_4$ the agent dropped a ball for the last time. After identifying those milestone time steps, the SRM rewards the behaviors at the corresponding time steps. The intuition behind such design pattern is that the reward function simply encourages all those behaviors if it is unclear what outcome those behavior will lead to; once the agent unlocks a door, the reward function is able to identify the milestone behaviors that are most closely related to the door unlocking outcome. Then we show the atomic predicates in the symbolic constraint for this task in Table.3. The final symbolic constraint is $c = \bigwedge_{i=1}^{12} \mu_i$.

Note that the stored reward is not to be confused with the reward output at the present time. The syntax of sequencing in the `Hindsignt` code block depends on the language of the r term specified in the background theory. For the other two SRMs annotated by SRM2 and SRM3 as mentioned earlier, we remove the `Hindsight` block. Especially, in SRM2, we restrict that door unlocking and door opening behaviors are rewarded if only the accumulated rewards gained from those two behaviors do not exceed $?_2$. Otherwise, none of the behaviors correlated with the self-looping transitions at state "`Before_Seeing_the_Target`" in Fig.10 will ever be rewarded. A possible reason for the policies trained by SRM1 do not generalize well in larger environment is that due to the hindsight reward modification, the reward output is too sparse in the large environment for the agent to learn. As shown by the experimental results of SRM2 and SRM3, once the `Hindsight` block is removed, the training performance in large environment is improved.

Properties	(Retaional) Predicates	(Non-Relational) Predicates
$[\mu_1]$Reward picking up target	$\bigwedge_{\texttt{id}=2}^{12}(?_{\texttt{id}} \leq ?_1)$	$?_1 \geq 0$
$[\mu_2]$Reward finding target	$?_2 \geq ?_4 + ?_5 - 2?_3$	$?_2 \geq 0$
$[\mu_3]$Reward opening door	$?_3 \leq 0$	$?_3 \leq 0$
$[\mu_4]$Reward opening door	$?_4 \geq 0$	$?_4 \geq 0$
$[\mu_5]$Reward unlocking door	$?_5 \geq \sum_{\texttt{id}=6}^{10} ?_{\texttt{id}}$	$?_5 \geq 0$
$[\mu_6]$Penalize meaningless move	$?_3 \leq 0$	$?_3 \geq 0$
$[\mu_7]$Penalize picking up used key	$?_{11} + ?_{12} \leq 0$	$?_{11} \leq 0$
$[\mu_8]$Reward opening box	$?_6 \geq 0$	$?_6 \geq 0$
$[\mu_9]$Reward picking up ball	$?_7 \geq 0$	$?_7 \geq 0$
$[\mu_10]$Reward picking up key	$?_8 \geq 0$	$?_8 \geq 0$
$[\mu_{11}]$Reward dropping ball	$?_9 \geq 0$	$?_9 \geq 0$
$[\mu_{12}]$Reward dropping used key	$?_{12} \geq 0$	$?_{12} \geq 0$

Table 3: The correspondence between properties and predicates for the SRM of ObstructedMaze task in Fig.10

A.3 Training details

- **Training Overhead**. We note that all the designed SRMs require checking hindsight experiences, or maintaining memory or other expensive procedures. However, line 5 of Algorithm 1 requires running all K candidate programs on all m sampled trajectories, which may incur a substantial overhead during training. Our solution is that, before sampling any program as in line 5 of Algorithm 1, we evaluate the result of $[\![\mathcal{L}]\!](\tau_{A,i})$, which keeps holes **?** unassigned, for all the m trajectories. By doing this, we only need to execute the expensive procedures that do not involve the holes once, such as the counter #OPENDOOR and the reward modification steps in the Hindsight block in Fig.10. Then we use q_φ to sample K hole assignments $\{\mathbf{h_k}\}_{k=1}^K$ from $\mathbf{H}$ and feed them to $\{[\![\mathcal{L}]\!](\tau_{A,i})\}_{i=1}^m$ to obtain $\{\{[\![l_k := \mathcal{L}[\mathbf{h}_k/\mathbf{?}]]\!](\tau_{A,i})\}_{i=1}^m\}_{k=1}^K$. By replacing line 2 and line 5 with those two steps in Algorithm 1, we significantly reduce the overhead.

- **Supervised Learning Loss**. In Algorithm 1, a supervised learning objective J_{cons} is used to penalize any sampled hole assignment for not satisfying the symbolic constraint. In practice, since our sampler q_φ directly outputs the mean and log-variance of a multivariate Gaussian distribution for the candidate hole assignments, we directly evaluate the satisfaction of the mean. Besides, as mentioned earlier, in our experiments we only consider symbolic constraint as a conjunction of atomic predicates, e.g., $c = \wedge_{i=1}^n \mu_i$ with each μ_i only concerning linear combinations of the holes, we reformulated each μ_i into a form $u_i(\mathbf{?}) \leq 0$ where u_i is some linear function of the holes **?**. We make sure that $(u_i(\mathbf{h}) \leq 0) \leftrightarrow ([\![\mu_i]\!](\mathbf{h}) = \top)$ for any hole assignment $\mathbf{h}$. After calculating each $u_i(\mathbf{h})$, which is now a real number, we let $J_{cons}(q_\varphi)$ be a negative binary cross-entropy loss for $Sigmoid(ReLU([u_i(\mathbf{h}), \ldots, u_n(\mathbf{h})]^T)))$ with 0 being the ground truth. This loss penalizes any $\mathbf{h}$ that makes $u_i(\mathbf{h}) > 0$. In this way $J_{cons}(q_\varphi)$ is differentiable w.r.t φ. Besides, we retain the entropy term $\mathcal{H}(q_\varphi)$ extracted from the KL-divergence to regularize the variance output by q_φ.

  Network Architectures. Algorithm 1 involves an agent policy π_ϕ, a neural reward function f_θ and a sampler q_φ. Each of the three is composed of one or more neural networks.

  - **Agent policy** π_ϕ. We prepare two versions of actor-critic networks, a CNN version and an LSTM version. For the CNN version, we adopt the actor-critic network from the off-the-shelf implementation of AGAC Flet-Berliac et al. (2021). It has 3 convolutional layers each with 32 filters, 3×3 kernel size, and a stride of 2. A diagram of the CNN layers can be found in Flet-Berliac et al. (2021). For the LSTM version, we concatenate 3 identically configured convolutional layers with a LSTM cell of 32-size state vector. The LSTM cell is then followed by multiple fully connected layers each to simulate the policy, value and advantage functions. While AGAC contains other

Parameter	**Value**
# Epochs	4
# minibatches (π_ϕ)	8
# batch size (f_θ, q_φ)	128
# frames stacked (CNN π_ϕ)	4
# reccurence (LSTM π_ϕ)	1
# recurrence (f_θ)	8
Discount factor γ	0.99
GAE parameter λ	0.95
PPO clipping parameter ϵ	0.2
K	16
α	0.001
β	0.0003
η	1.e8
$Entropy$	1.e-2

Table 4: Hyperparameters used in the training processes

components Flet-Berliac et al. (2021), the PPO agent solely consists of the actor-critic networks.

– **Neural reward function** f_θ. The network is recurrent. It has 3 convolutional layers each with 16, 32 and 64 filters, 2×2 kernel size and a stride of 1. The last convolutional layer is concatenated with an LSTM cell of which the state vector has a size of 128. The LSTM cell is then followed by a 3-layer fully connected network where each hidden layer is of size 64. Between each hidden layer we use two $tanh$ functions and one Sigmoid function as the activation functions. The output of the Sigmoid function is the logit for each action in the action space $\mathcal{A}$. Finally, given an action in a state, we use softmax and a Categorical distribution output the log-likelihood for the given action as the reward.

– **Sampler** q_φ. The input to q_φ is a constant $[1, \ldots, 1]^T$ of size 20. The sampler is a fully-connected network with 2 hidden layers of size 64. The activation functions are both $tanh$. Suppose that there are $|\mathbf{?}|$ holes in the SRM. Then the output of q_φ is a vector of size no less than $2|\mathbf{?}|$. The $|\mathbf{?}|$ most and the $|\mathbf{?}|$ least significant elements in the output vector will respectively be used as the mean of the Gaussian and constitute a diagonal log-variance matrix. Besides, we let q_φ to output a value as the constant reward for the dummy transitions. While we still return 0 instead of this constant as the reward to the agent, we subtract every sampled $\mathbf{h}$ with this constant to compute $[\![l]\!](\tau)$ for J_{soft}. This subtraction simulates normalizing $[\![l]\!](\tau)$ in order to match the outputs of f_θ, which, as mentioned earlier, is always non-positive in order to match $\log \pi_E$.

- **Hyperparameters**. Most of the hyperparameters that appear in Algorithm 1 are summarized as in Table.4. All hyperparameters relevant to AGAC are identical as those in Flet-Berliac et al. (2021) although we do not present all of them in Table.4 in order to avoid confusion. The hyperparameter η is made large to heavily penalize q_φ when its output violates the symbolic constraint c. Besides, we add an entropy term $\mathcal{H}(q_\varphi)$ multiplied by $1e-2$ in addition to J_{soft} and J_{con} to regularize the variance output by q_φ. The item $Entropy$ refers to the multiplier for the entropy term $\mathcal{H}(q_\varphi)$ as introduced earlier.

A.4 Derivation Of the Objective Functions

First, we derive the lower-bound of $\log p(0_A, 1_E|\pi_A, E, l)$ in Eq.5 as follows.

$$
\begin{aligned}
& \log p(0_A, 1_E|\pi_A, E, l) \\
= \; & \log \sum_{\tau_A, \tau_E} p(\tau_A|\pi_A)p(\tau_E|E) \iint\limits_{f_{\tau_A}, f_{\tau_E}} p(0_A|\tau_A; \pi_A, f_{\tau_A})p(1_E|\tau_E; \pi_A, f_{\tau_E}) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad p(f_{\tau_E}|\tau_E; l)p(f_{\tau_A}|\tau_A; l) \\
\geq \; & \mathop{\mathbb{E}}_{\substack{\tau_A \sim \pi_A \\ \tau_E \sim E}} \Big[\log \iint\limits_{f_{\tau_A}, f_{\tau_E}} p(0_A|\tau_A; \pi_A, f_{\tau_A})p(1_E|\tau_E; \pi_A, f_{\tau_E})p(f_{\tau_E}|\tau_E; l)p(f_{\tau_A}|\tau_A; l) \Big] \\
= \; & \mathop{\mathbb{E}}_{\substack{\tau_A \sim \pi_A \\ \tau_E \sim E}} \Big[\log \iint\limits_{f_{\tau_A}, f_{\tau_E}} p(0_A|\tau_A; \pi_A, f_{\tau_A})p(1_E|\tau_E; \pi_A, f_{\tau_E})p(f_{\tau_A}|\tau_A; l)p(f_{\tau_E}|\tau_E; l) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \frac{p(f_{\tau_A}|\tau_A; f)p(f_{\tau_E}|\tau_E; f)}{p(f_{\tau_A}|\tau_A; f)p(f_{\tau_E}|\tau_E; f)} \Big] \\
\geq \; & \max_f \mathop{\mathbb{E}}_{\substack{\tau_A \sim \pi_A \\ \tau_E \sim E}} \Big[\mathop{\mathbb{E}}_{\substack{f_{\tau_A} \sim p(\cdot|\tau_A; f) \\ f_{\tau_E} \sim p(\cdot|\tau_E; f))}} \Big(\log p(0_A|\tau_A; \pi_A, f_{\tau_A})p(1_E|\tau_E; \pi_A, f_{\tau_E}) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \frac{p(f_{\tau_A}|\tau_A; l)p(f_{\tau_E}|\tau_E; l)}{p(f_{\tau_A}|\tau_A; f)p(f_{\tau_E}|\tau_E; f)} \Big) \Big] \\
= \; & \max_f \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{N}} \Big[J_{adv}(D_\epsilon) \Big] - \mathop{\mathbb{E}}_{\tau \sim \pi_A, E} \Big[D_{KL}(p_{f(\tau)} || p_{l(\tau)}) \Big]
\end{aligned}
$$

We justify the usage of the stochastic version $\mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)}[J_{adv}(D_\epsilon)]$, rather than the conventional generative adversarial objective $J_{adv}(D)$, by showing that one of the saddle point of $\min_{\pi_A} \max_f \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)}[J_{adv}(D_\epsilon)]$ is attained when $f \equiv \log \pi_E \equiv \log \pi_A$ where we write π_E in proxy of E by assuming that the distribution of state-action pairs satisfies $p(s, a|\pi_E) \equiv p(s, a|E)$.

Theorem 1. *Given a* π_E, $\min_{\pi_A} \max_f \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \big\{ \mathbb{E}_{(s,a) \sim \pi_E} \big[\log D_\epsilon(s, a) \big] + \mathbb{E}_{(s,a) \sim \pi_A} \big[\log(1 - D_\epsilon(s, a) \big] \big\}$, *where* $D_\epsilon(s, a) := \frac{\exp(f(s,a)+\epsilon)}{\exp(f(s,a)+\epsilon)+\pi_A(a|s))}$, *is optimal when* $f \equiv \log \pi_E \equiv \log \pi_A$.

Proof. Firstly, we consider optimizing f under the condition of $\pi_A \equiv \pi_E$. Inspired by the proof of optimality condition of Generative Adversarial Nets in Goodfellow et al. (2014), we introduce two variables $x_{s,a} = y_{s,a} \in (0, 1]$ to simulate $p(s, a|\pi_A) = p(s, a|\pi_E)$ for any $s, a \in \mathcal{S} \times \mathcal{A}$. Then we prove that $\mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \big[x_{s,a} \log \frac{\hat{x}_{s,a} \cdot \exp(\epsilon)}{\hat{x}_{s,a} \cdot \exp(\epsilon) + y_{s,a}} + y_{s,a} \log \frac{y_{s,a}}{\hat{x}_{s,a} \cdot \exp(\epsilon) + y_{s,a}} \big]$ as a function of $\hat{x}_{s,a}$ has x stationary point at $\hat{x}_{s,a} = x_{s,a}$ by computing its gradient w.r.t $\hat{x}_{s,a}$ as follows.

$$
\begin{aligned}
& \nabla_{\hat{x}_{s,a}} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \big[x_{s,a} \log \frac{\hat{x}_{s,a} \cdot \exp(\epsilon)}{\hat{x}_{s,a} \cdot \exp(\epsilon) + y_{s,a}} + y_{s,a} \log \frac{y_{s,a}}{\hat{x}_{s,a} \cdot \exp(\epsilon) + y_{s,a}} \big] \\
= \; & \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \big[x_{s,a} \cdot \frac{\hat{x}_{s,a} \cdot \exp(\epsilon) + y_{s,a}}{\hat{x}_{s,a} \cdot \exp(\epsilon)} \cdot \frac{\exp(\epsilon)(\hat{x}_{s,a} \cdot \exp(\epsilon) + y_{s,a}) - \hat{x}_{s,a} \exp(2\epsilon)}{(\hat{x}_{s,a} \cdot \exp(\epsilon) + y_{s,a})^2} + \\
& \qquad\qquad y_{s,a} \cdot \frac{\hat{x}_{s,a} \cdot \exp(\epsilon) + y_{s,a}}{y_{s,a}} \cdot \frac{-y_{s,a} \exp(\epsilon)}{(\hat{x}_{s,a} \cdot \exp(\epsilon) + y_{s,a})^2} \big] \\
= \; & \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \big[\frac{x_{s,a} y_{s,a} / \hat{x}_{s,a} - y_{s,a} \exp(\epsilon)}{\hat{x}_{s,a} \cdot \exp(\epsilon) + y_{s,a}} \big] \qquad (7)
\end{aligned}
$$

When $\hat{x}_{s,a} = x_{s,a} = y_{s,a}$, Eq.7 equals $\mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \big[\frac{1-\exp(\epsilon)}{1+\exp(\epsilon)} \big]$. Note that the probabilities of sampling ϵ and $-\epsilon$ equal each other, and $\frac{1-\exp(\epsilon)}{1+\exp(\epsilon)} = -\frac{1-\exp(-\epsilon)}{1+\exp(-\epsilon)}$. Hence, $\mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \big[\frac{1-\exp(\epsilon)}{1+\exp(\epsilon)} \big] = 0$. It can trivially proved that the gradient of Eq.7 w.r.t $\hat{x}$ is non-positive. Therefore, $f \equiv \log \pi_E$ is a local maximum.

Next, we consider optimizing π_A under the condition of $f \equiv \log \pi_E$. We denote $p(s,a|\pi_E)$ and $p(s,a|\pi_A)$ for any $(s,a) \in \mathcal{S} \times \mathcal{A}$ as $x_{s,a}, y_{s,a} \in (0,1]$ for short. Then we show that $\mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)}\big[\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} x_{s,a} \log \frac{x_{s,a}\cdot\exp(\epsilon)}{x_{s,a}\cdot\exp(\epsilon)+y_{s,a}} + \frac{y_{s,a}}{x_{s,a}+y_{s,a}} \log \frac{y_{s,a}}{x_{s,a}\cdot\exp(\epsilon)+y_{s,a}}\big]$ as a function of $\{y_{s,a}|(s,a) \in \mathcal{S}\times\mathcal{A}\}$ s.t. $\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} y_{s,a} = 1$ has a stationary point at $x_{s,a} \equiv y_{s,a}$ by computing the gradient of the Lagrangian of this constrained function as follows.

$$
\begin{aligned}
L &= \mathop{\mathbb{E}}_{\epsilon\sim\mathcal{N}(0,1)}\Big[\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} x_{s,a} \log \frac{x_{s,a}\cdot\exp(\epsilon)}{x_{s,a}\cdot\exp(\epsilon)+y_{s,a}} + y_{s,a} \log \frac{y_{s,a}}{x_{s,a}\cdot\exp(\epsilon)+y_{s,a}}\Big] - \\
&\qquad\qquad \lambda\Big(\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} y_{s,a} - 1\Big) \\
\nabla_{y_{s,a}} L &= \mathop{\mathbb{E}}_{\epsilon\sim\mathcal{N}(0,1)}\Big[\log \frac{y_{s,a}}{x_{s,a}\cdot\exp(\epsilon)+y_{s,a}} + \frac{x_{s,a}(\exp(\epsilon)-1)}{x_{s,a}\cdot\exp(\epsilon)+y_{s,a}}\Big] - \lambda = 0 \\
&\Rightarrow \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}.\ \mathop{\mathbb{E}}_{\epsilon\sim\mathcal{N}(0,1)}\Big[\log \frac{y_{s,a}}{x_{s,a}\cdot\exp(\epsilon)+y_{s,a}} + \frac{x_{s,a}(\exp(\epsilon)-1)}{x_{s,a}\cdot\exp(\epsilon)+y_{s,a}}\Big] = \lambda \quad (8) \\
\nabla_\lambda L &= \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} y_{s,a} - 1 = 0 \quad (9)
\end{aligned}
$$

Suppose that $\lambda = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}[\log \frac{1}{1+\exp(\epsilon)}]$ and $x_{s,a} = y_{s,a}$ holds for any $(s,a) \in \mathcal{S}\times\mathcal{A}$. Then both Eq.8 and Eq.9 hold. Hence, $x_{s,a} \equiv y_{s,a}$ is a stationary point. It is also trivially provable that the gradient of Eq.8 w.r.t $y_{s,a}$ is non-negative. Therefore, $\pi_A \equiv \pi_E$ is a local minimum. In conclusion, $f \equiv \log \pi_E \equiv \log \pi_A$ is a saddle point. □

We derive the lower-bound of the ELBO in Eq.6 as follows.

$$
\begin{aligned}
& ELBO(q) = D_{KL}\Big[q(l)||p(l)\Big] + \mathop{\mathbb{E}}_{l\sim q}\Big[\log p(0_A, 1_E|\pi_A, E, l)\Big] \\
= \ & \mathop{\mathbb{E}}_{l\sim q}\Big\{\log \mathop{\mathbb{E}}_{\substack{\tau_A\sim\pi_A \\ \tau_E\sim E}}\Big[\iint_{f_{\tau_A}, f_{\tau_E}} p(0_A|\tau_A;\pi_A,f_{\tau_A}) p(1_E|\tau_E;\pi_A,f_{\tau_E}) p(f_{\tau_E}|\tau_E;l) p(f_{\tau_A}|\tau_A;l)\Big]\Big\} - \\
& \qquad\qquad D_{KL}\Big[q(l)||p(l)\Big] \\
\geq \ & \max_f \mathop{\mathbb{E}}_{\epsilon\sim\mathcal{N}}\Big[J_{adv}(D_\epsilon)\Big] - \mathop{\mathbb{E}}_{\substack{l\sim q \\ \tau\sim\pi_A, E}}\Big[D_{KL}(p_{f(\tau)}||p_{l(\tau)})\Big] - D_{KL}\Big[q(l)||p(l)\Big] \\
= \ & J_{soft}(q,f) + J_{con}(q)
\end{aligned}
$$