Fast Inference and Transfer of Compositional Task Structures for Few-shot Task Generalization

Sungryull Sohn[1, 2] **Hyunjae Woo**[1] **Jongwook Choi**[1] **Lyubing Qiang**[1] **Izzeddin Gur**[3] **Aleksandra Faust**[3]
Honglak Lee[1,2]

[1]University of Michigan
[2]LG AI Research Center Ann Arbor
[3]Google Research

1 DERIVATION OF SUBTASK GRAPH INFERENCE

Following Sohn et al. [2019], we formulate the problem of inferring the precondition $G_{\mathbf{c}}$ and the subtask reward $G_{\mathbf{r}}$ as a maximum likelihood estimation (MLE) problem. Let $\tau_K = \{\mathbf{s}_1, \mathbf{o}_1, r_1, d_1, \ldots, \mathbf{s}_K\}$ be an adaptation trajectory of the adaptation policy $\pi_\theta^{\text{adapt}}$ for K steps in adaptation phase. The goal is to infer the subtask graph G for this task, specified by preconditions $G_{\mathbf{c}}$ and subtask rewards $G_{\mathbf{r}}$. Consider a subtask graph G with subtask reward $G_{\mathbf{r}}$ and precondition $G_{\mathbf{c}}$. The maximum-likelihood estimate (MLE) of latent variables G, conditioned on the trajectory τ_H can be written as

$$\widehat{G}^{\text{MLE}} = \arg\max_{G_{\mathbf{c}}, G_{\mathbf{r}}} p(\tau_K | G_{\mathbf{c}}, G_{\mathbf{r}}). \tag{1}$$

The likelihood term can be expanded as

$$p(\tau_K | G_{\mathbf{c}}, G_{\mathbf{r}}) = p(\mathbf{s}_1 | G_{\mathbf{c}}) \prod_{t=1}^{K} \pi_\theta\left(\mathbf{o}_t | \tau_t\right) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{o}_t, G_{\mathbf{c}}) p(r_t | \mathbf{s}_t, \mathbf{o}_t, G_{\mathbf{r}}) p(d_t | \mathbf{s}_t, \mathbf{o}_t) \tag{2}$$

$$\propto p(\mathbf{s}_1 | G_{\mathbf{c}}) \prod_{t=1}^{K} p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{o}_t, G_{\mathbf{c}}) p(r_t | \mathbf{s}_t, \mathbf{o}_t, G_{\mathbf{r}}), \tag{3}$$

where we dropped the terms that are independent of G. From the definitions in **Subtask Graph Inference Problem** section, precondition $G_{\mathbf{c}}$ defines the mapping $\mathbf{x} \mapsto \mathbf{e}$, and the subtask reward $G_{\mathbf{r}}$ determines the reward as $r_t \sim {G_{\mathbf{r}}}^i$ if subtask i is eligible (*i.e.*, $\mathbf{e}_t^i = 1$) and option $\mathcal{O}^i$ is executed at time t. Therefore, we have

$$\widehat{G}^{\text{MLE}} = (\widehat{G}_{\mathbf{c}}^{\text{MLE}}, \widehat{G}_{\mathbf{r}}^{\text{MLE}}) = \left(\arg\max_{G_{\mathbf{c}}} \prod_{t=1}^{K} p(\mathbf{e}_t | \mathbf{x}_t, G_{\mathbf{c}}),\ \arg\max_{G_{\mathbf{r}}} \prod_{t=1}^{K} p(r_t | \mathbf{e}_t, \mathbf{o}_t, G_{\mathbf{r}}) \right). \tag{4}$$

Below we explain how to compute the estimate of preconditions $\widehat{G}_{\mathbf{c}}^{\text{MLE}}$ and subtask rewards $\widehat{G}_{\mathbf{r}}^{\text{MLE}}$.

Precondition inference via logic induction From the definition in **Subtask Graph Inference Problem** section, the mapping from a precondition $G_{\mathbf{c}}$ and a completion vector $\mathbf{x}$ to an eligibility vector $\mathbf{e} = f_{G_{\mathbf{c}}}(\mathbf{x})$ is a deterministic function (*i.e.*, precondition function). Therefore, we can rewrite $\widehat{G}_{\mathbf{c}}^{\text{MLE}}$ in Eq.(4) as:

$$\widehat{G}_{\mathbf{c}}^{\text{MLE}} = \arg\max_{G_{\mathbf{c}}} \prod_{t=1}^{K} \mathbb{I}(\mathbf{e}_t = f_{G_{\mathbf{c}}}(\mathbf{x}_t)), \tag{5}$$

where $\mathbb{I}(\cdot)$ is the indicator function. Since the eligibility $\mathbf{e}$ is factored, the precondition function $f_{G_{\mathbf{c}}{}^i}$ for each subtask is inferred independently. This problem of finding a boolean function that satisfies all the indicator functions in Eq.(5) (*i.e.*, $\prod_{t=1}^{K} \mathbb{I}(\mathbf{e}_t = f_{G_{\mathbf{c}}}(\mathbf{x}_t)) = 1$) is formulated as an *inductive logic programming* (ILP) problem [Muggleton, 1991]. Specifically, $\{\mathbf{x}_t\}_{t=1}^K$ forms binary vector inputs to programs, and $\{e_t^i\}_{t=1}^K$ forms Boolean-valued outputs of the i-th program

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

that denotes the eligibility of the i-th subtask. We use the *classification and regression tree* (CART) to infer the precondition function $f_{G_\mathbf{c}}$ for each subtask based on Gini impurity [Breiman, 1984]. Intuitively, the constructed decision tree is the simplest boolean function approximation for the given input-output pairs $\{\mathbf{x}_t, \mathbf{e}_t\}$. Then, we convert it to a logic expression (*i.e.*, precondition) in sum-of-product (SOP) form to build the subtask graph.

Subtask reward inference To infer the subtask reward function $\widehat{G}_\mathbf{r}^{\text{MLE}}$ in Eq.(4), we model each component of subtask reward as a Gaussian distribution $G_\mathbf{r}^i \sim \mathcal{N}(\widehat{\mu}^i, \widehat{\sigma}^i)$. Then, $\widehat{\mu}^i_{\text{MLE}}$ becomes the empirical mean of the rewards received after taking the eligible option $\mathcal{O}^i$ in the trajectory τ_K:

$$\widehat{G}_\mathbf{r}^{\text{MLE},i} = \widehat{\mu}^i_{\text{MLE}} = \mathbb{E}\left[r_t | \mathbf{o}_t = \mathcal{O}^i, \mathbf{e}_t^i = 1\right] = \frac{\sum_{t=1}^K r_t \mathbb{I}(\mathbf{o}_t = \mathcal{O}^i, \mathbf{e}_t^i = 1)}{\sum_{t=1}^K \mathbb{I}(\mathbf{o}_t = \mathcal{O}^i, \mathbf{e}_t^i = 1)}. \tag{6}$$

2 DETAILS OF GRPROP POLICY

For self-containedness, we repeat the description of GRProp policy in Sohn et al. [2019].

Intuitively, GRProp policy modifies the subtask graph to a differentiable form such that we can compute the gradient of modified return with respect to the subtask completion vector in order to measure how much each subtask is likely to increase the modified return. Let $\mathbf{x}_t$ be a completion vector and $G_{\mathbf{r}}$ be a subtask reward vector (see **Subtask Graph Inference Problem** section for definitions). Then, the sum of reward until time-step t is given as:

$$U_t = {G_{\mathbf{r}}}^\top \mathbf{x}_t. \tag{7}$$

We first modify the reward formulation such that it gives a half of subtask reward for satisfying the preconditions and the rest for executing the subtask to encourage the agent to satisfy the precondition of a subtask with a large reward:

$$\widehat{U}_t = {G_{\mathbf{r}}}^\top (\mathbf{x}_t + \mathbf{e}_t)/2. \tag{8}$$

Let y^j_{AND} be the output of j-th `AND` node. The eligibility vector $\mathbf{e}_t$ can be computed from the subtask graph G and $\mathbf{x}_t$ as follows:

$$e^i_t = \underset{j \in Child_i}{\text{OR}} \left(y^j_{AND} \right), \quad y^j_{AND} = \underset{k \in Child_j}{\text{AND}} \left(\widehat{x}^{j,k}_t \right), \quad \widehat{x}^{j,k}_t = x^k_t w^{j,k} + \text{NOT}(x^k_t)(1 - w^{j,k}), \tag{9}$$

where $w^{j,k} = 0$ if there is a `NOT` connection between j-th node and k-th node, otherwise $w^{j,k} = 1$. Intuitively, $\widehat{x}^{j,k}_t = 1$ when k-th node does not violate the precondition of j-th node. The logical AND, OR, and NOT operations in Equation (9) are substituted by the smoothed counterparts as follows:

$$p^i = \lambda_{\text{or}} \widetilde{e}^i + (1 - \lambda_{\text{or}})\, x^i, \tag{10}$$

$$\widetilde{e}^i = \underset{j \in Child_i}{\widetilde{\text{OR}}} \left(\widetilde{y}^j_{AND} \right), \tag{11}$$

$$\widetilde{y}^j_{AND} = \underset{k \in Child_j}{\widetilde{\text{AND}}} \left(\hat{x}^{j,k} \right), \tag{12}$$

$$\hat{x}^{j,k} = w^{j,k} p^k + (1 - w^{j,k}) \widetilde{\text{NOT}} \left(p^k \right), \tag{13}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the input completion vector,

$$\widetilde{\text{OR}}\,(\mathbf{x}) = \text{softmax}(w_{\text{or}} \mathbf{x}) \cdot \mathbf{x}, \tag{14}$$

$$\widetilde{\text{AND}}\,(\mathbf{x}) = \frac{\zeta(\mathbf{x}, w_{\text{and}})}{\zeta(||\mathbf{x}||, w_{\text{and}})}, \tag{15}$$

$$\widetilde{\text{NOT}}\,(\mathbf{x}) = -w_{\text{not}} \mathbf{x}, \tag{16}$$

$||\mathbf{x}|| = d$, $\zeta(\mathbf{x}, \beta) = \frac{1}{\beta} \log(1 + \exp(\beta \mathbf{x}))$ is a soft-plus function, and $\lambda_{\text{or}} = 0.6, w_{\text{or}} = 2, w_{\text{and}} = 3, w_{\text{not}} = 2$ are the hyper-parameters of GRProp. Note that we slightly modified the implementation of $\widetilde{\text{OR}}$ and $\widetilde{\text{AND}}$ from sigmoid and hyper-tangent functions in [Sohn et al., 2018] to softmax and softplus functions for better performance. With the smoothed operations, the sum of smoothed and modified reward is given as:

$$\widetilde{U}_t = {G_{\mathbf{r}}}^\top \mathbf{p}, \tag{17}$$

where $\mathbf{p} = [p^1, \ldots, p^d]$ and p^i is computed from Equation (10). Finally, the graph reward propagation policy is a softmax policy,

$$\pi(\mathbf{o}_t | G, \mathbf{x}_t) = \text{Softmax} \left(T \nabla_{\mathbf{x}_t} \widetilde{U}_t \right) = \text{Softmax} \left(T {G_{\mathbf{r}}}^\top (\lambda_{\text{or}} \nabla_{\mathbf{x}_t} \widetilde{\mathbf{e}}_t + (1 - \lambda_{\text{or}})) \right), \tag{18}$$

where we used the softmax temperature $T = 40$ for **Playground** and **Mining** domain, and linearly annealed the temperature from $T = 1$ to $T = 40$ during adaptation phase for **SC2LE** domain. Intuitively speaking, we act more confidently (*i.e.*, higher temperature T) as we collect more data since the inferred subtask graph will become more accurate.

3 DERIVATION OF MULTI-TASK SUBTASK GRAPH INFERENCE

Let τ be the adaptation trajectory of the current task $\mathcal{M}_G$, and $\mathcal{T}^{\mathrm{p}} = \{\tau_1^{\mathrm{p}}, \ldots, \tau_{|\mathcal{T}^{\mathrm{p}}|}^{\mathrm{p}}\}$ be the adaptation trajectories of the seen training tasks.

Then, from Bayesian rule, we have

$$\begin{aligned}
\pi(o|s, \tau, \mathcal{T}^{\mathrm{p}}) &= \textstyle\sum_G \pi(o|s, G) p(G|\tau, \mathcal{T}^{\mathrm{p}}) && (19)\\
&\propto \textstyle\sum_G \pi(o|s, G) p(\tau|G, \mathcal{T}^{\mathrm{p}}) p(G|\mathcal{T}^{\mathrm{p}}) && (20)\\
&= \textstyle\sum_G \pi(o|s, G) p(\tau|G) p(G|\mathcal{T}^{\mathrm{p}}) && (21)\\
&\propto \textstyle\sum_G \pi(o|s, G) p(\tau|G) p(\mathcal{T}^{\mathrm{p}}|G) p(G), && (22)
\end{aligned}$$

where Equation (21) holds because τ and $\mathcal{T}^{\mathrm{p}}$ are independently observed variables. Since summing over all G is computationally intractable, we instead approximate it by computing the sample estimates of π. Specifically, we compute the policy $\pi(o|s, G)$ at the maximum likelihood estimates (MLE) of G for prior and current tasks, that is $G^\tau = \arg\max_G p(\tau|G)$ and $G^{\mathrm{p}} = \arg\max_G p(\mathcal{T}^{\mathrm{p}}|G)$ respectively, and combine them with the weight α:

$$\pi(o|s, \tau, \mathcal{T}^{\mathrm{p}}) \simeq \pi(o|s, G^\tau)^\alpha \pi(o|s, G^{\mathrm{p}})^{(1-\alpha)}. \tag{23}$$

Finally, we deploy the GRProp policy as a contextual policy:

$$\pi^{\text{eval}}(\cdot|\tau, \mathcal{T}^{\mathrm{p}}) \simeq \text{GRProp}(\cdot|G^\tau)^\alpha \text{GRProp}(\cdot|G^{\mathrm{p}})^{(1-\alpha)}. \tag{24}$$

4 PSEUDO-CODE OF OUR ALGORITHM

The Algorithm 1 below describes the pseudo-code of the meta-training process of our algorithm.

Algorithm 1 Meta-training: learning the prior

Require: Adaptation policy π^{adapt}
Ensure: Prior set $\mathcal{T}^{\mathrm{p}}$
1: $\mathcal{T}^{\mathrm{p}} \leftarrow \emptyset$
2: **for** each task $\mathcal{M} \in \mathcal{M}^{\text{train}}$ **do**
3: Rollout adaptation policy:
$\tau = \{\mathbf{s}_t, \mathbf{o}_t, r_t, d_t\}_{t=1}^K \sim \pi^{\text{adapt}}$ in task $\mathcal{M}$
4: Infer subtask graph $G^\tau = \arg\max_G p(\tau|G)$
5: $\pi^{\text{eval}} = \text{GRProp}(G^\tau)$
6: Evaluate the agent: $\tau^{\text{eval}} \sim \pi^{\text{eval}}$ in task $\mathcal{M}$
7: Update prior $\mathcal{T}^{\mathrm{p}} \leftarrow \mathcal{T}^{\mathrm{p}} \cup (G^\tau, \tau)$
8: **end for**

5 EXTENDED RELATED WORK

Multi-task reinforcement learning. Multi-task reinforcement learning aims to learn an inductive bias that can be shared and used across a variety of related RL tasks to improve the task generalization. Early works mostly focused on the transfer learning oriented approaches [Lazaric, 2012, Taylor and Stone, 2009] such as instance transfer [Lazaric et al., 2008] or representation transfer [Konidaris and Barto, 2006]. However, these algorithms rely heavily on the prior knowledge about the allowed task differences. Hausman et al. [2018], Pinto and Gupta [2017], Wilson et al. [2007] proposed to train a multi-task policy with multiple objectives from different tasks. However, the gradients from different tasks may conflict and hurt the training of other tasks. To avoid gradient conflict, Zhang and Yeung [2014], Chen et al. [2018], Lin et al. [2019] proposed to explicitly model the task similarity. However, dynamically modulating the loss or the gradient of RL update often results in the instability in optimization. Our multi-task learning algorithm also takes the transfer learning oriented viewpoint; MTSGI captures and transfers the task knowledge in terms of the subtask graph. However, our work does not make a strong assumption on the task distribution. We only assume that the task is parameterized by unknown subtask graph, which subsumes many existing compositional tasks (*e.g.*, Oh et al. [2017], Andreas et al. [2017], Huang et al. [2018], etc).

Extended - web navigating RL agent. Previous work introduced MiniWoB [Shi et al., 2017] and MiniWoB++ [Liu et al., 2018] benchmarks that are manually curated sets of simulated toy environments for the web navigation problem. They formulated the problem as acting on a page represented as a Document Object Model (DOM), a hierarchy of objects in the page. The agent is trained with human demonstrations and online episodes in an RL loop. Jia et al. [2019] proposed a graph neural network based DOM encoder and a multi-task formulation of the problem similar to this work. Gur et al. [2018] introduced a manually-designed curriculum learning method and an LSTM based DOM encoder. DOM level representations of web pages pose a significant sim-to-real gap as simulated websites are considerably smaller (100s of nodes) compared to noisy real websites (1000s of nodes). As a result, these models are trained and evaluated on the same simulated environments which is difficult to deploy on real websites. Our work formulates the problem as abstract web navigation on real websites where the objective is to learn a latent subtask dependency graph similar to sitemap of websites. We propose a multi-task training objective that generalizes from a fixed set of real websites to unseen websites without any demonstration, illustrating an agent capable of navigating real websites for the first time.

Planning Approaches for Compositional Task Previous work has tackled the compositional tasks using the Hierarchical Task Network (HTN) planning [Sacerdoti, 1975b,a, Tate, 1977] in a (single) goal-conditioned RL setting. The HTN allows the agent to reason the tasks at multiple levels of abstraction, when rich knowledge at those abstraction levels are available. Specifically, HTN models the primitive tasks (or the *subtasks* in our terminology) by the precondition and the effects, and aim to find the sequence of actions (or the *options* in our terminology) that execute each subtasks via planning on the HTN. They aim to execute a single goal task, often with assumptions of simpler subtask dependency structures [Ghazanfari and Taylor, 2017, Liu et al., 2016] such that the task structure can be constructed from the successful trajectories. Also, they often require expensive searching to find the solution. In contrast, we tackle a more general and challenging setting, where each subtask gives a reward (*i.e.*, multi-goal setting) and the goal is to maximize the cumulative sum of reward within an episode. Moreover, we avoid any expensive searching and propose to use neural network to directly map the task structure into policy. Lastly, we aim to achieve zero-/few-shot task generalization, which is not achievable with the HTN methods since they require the full specification of the action models in the testing.

6 ADDITIONAL EXPERIMENT RESULTS

6.1 FULL EXPERIMENT RESULTS ON THE PERFORMANCE OF THE AGENTS ON 15 WEBSITES IN *SYMWOB*

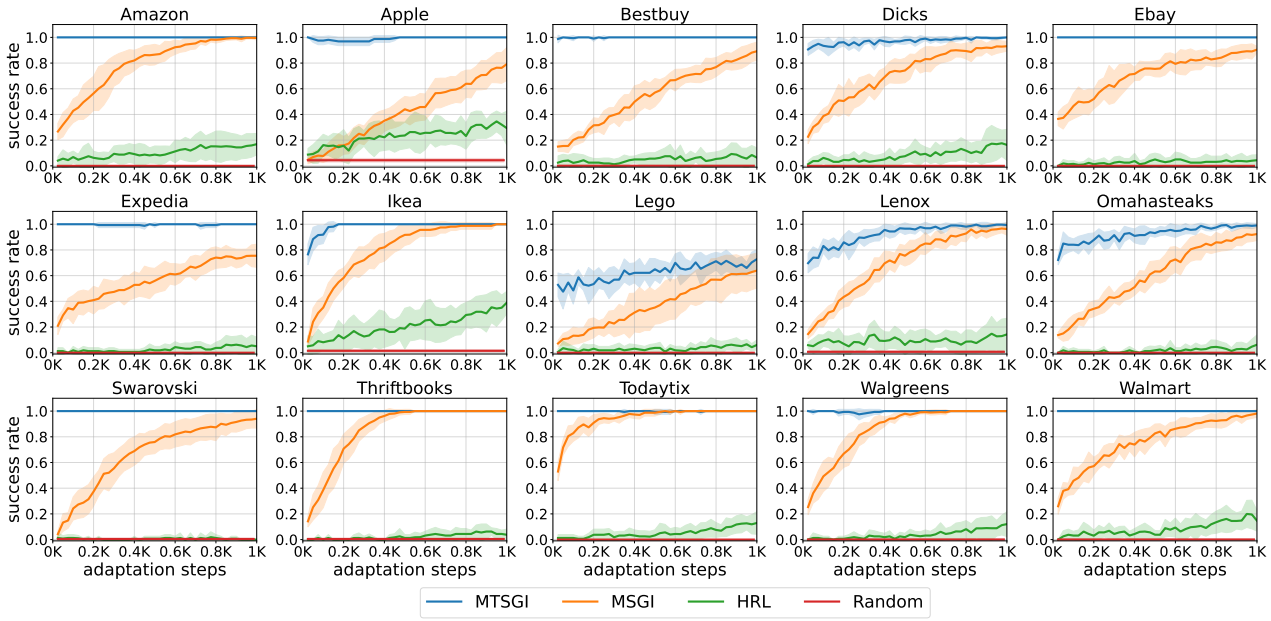

Figure 1: Results of the success rate for compared methods in the test phase with respect to the adaptation steps on 15 environments in *SymWoB* domain.

6.2 VISUALIZATION OF THE MULTI-TASK SUBTASK GRAPH INFERENCE PROCESS

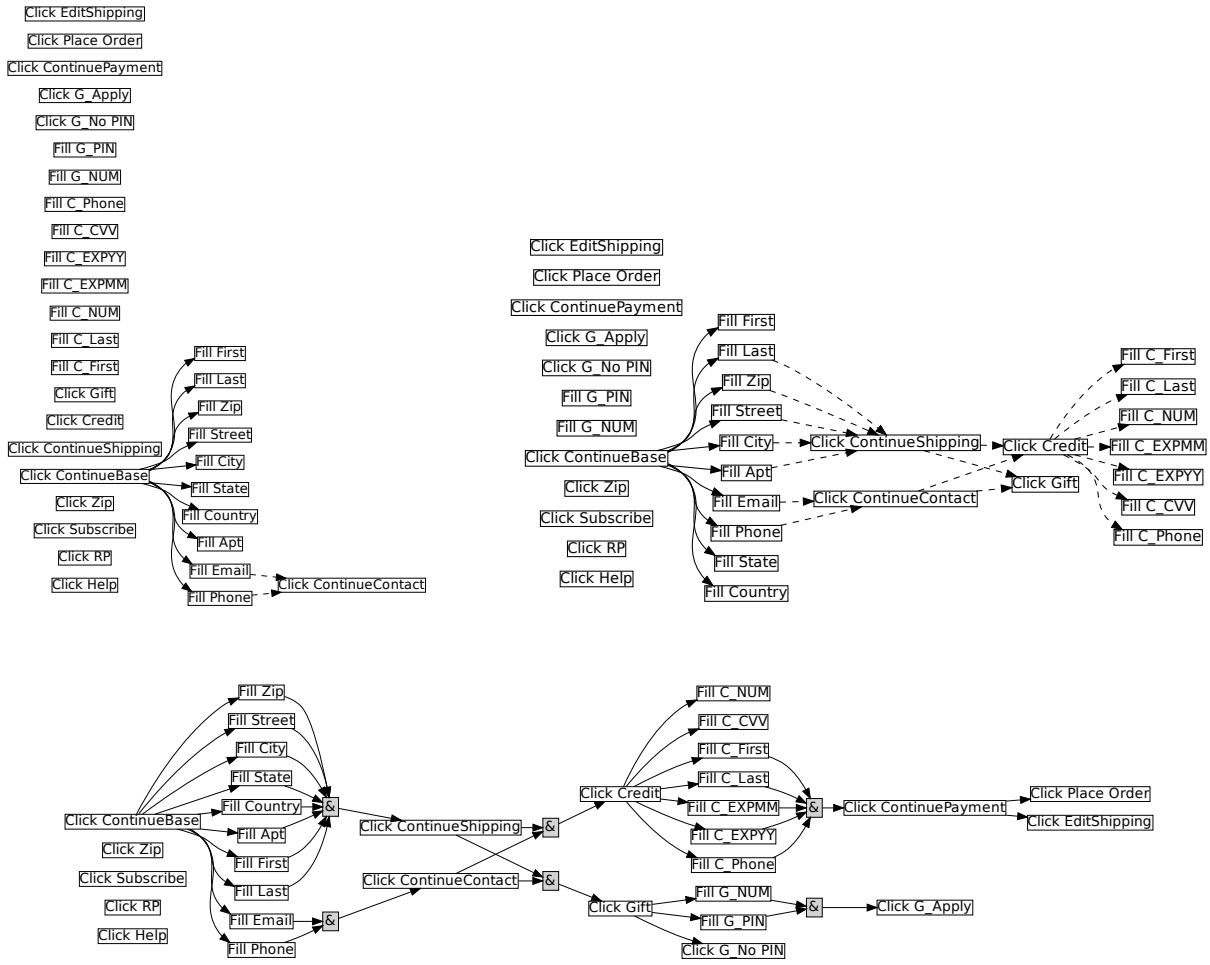

Figure 2: The subtask graphs inferred by our MTSGI with varying adaptation steps on **Walmart** domain: 0 steps (*Top, Left*), 400 steps (*Top, Right*), and 1000 steps (*Bottom*).

Figure 2 qualitatively shows how the multi-task subtask graph inference proceeds over the adaptation. In the beginning (*Top, Left*), the agent has only prior information and the prior provides a partial information about the subtask graph, which is the preconditions in the first webpage. As the agent further explores the webpage (*Top, Right*), the subtask graph gets more accurate, but due to insufficient exploration, there are many missing preconditions, especially for the subtasks in the last webpage. After sufficient adaptation (*Bottom*), our MTSGI can infer quite accurate subtask graph compared to the ground-truth subtask graph in Figure 3; *i.e.*, only missing two preconditions.

6.3 FULL EXPERIMENT RESULTS ON THE QUALITATIVE EVALUATION OF THE INFERRED SUBTASK GRAPH

Figures 3-17 qualitatively evaluate the task inference of our MTSGI by comparing the inferred subtask graph and the ground-truth. It is clear from the figure that the inferred subtask graphs have only a small portion of missing or redundant edges, while most of the nodes and edges are the same as ground-truth graph.

7 DETAILS ON *SYMWOB* DOMAIN

Subtask Graph Setting								
Task	**Amazon**	**Apple**	**BestBuy**	**Dick's**	**eBay**	**Expedia**	**Ikea**	**Lego**
#Subtasks	31	43	37	39	39	36	39	45
#Distractors	4	5	6	6	5	5	5	6
Episode length	27	40	37	37	37	40	37	37
Task	**Lenox**	**Omahasteaks**	**Swarovski**	**Thriftbooks**	**Todaytix**	**Walgreens**	**Walmart**	
#Subtasks	45	44	45	43	23	38	46	
#Distractors	4	6	7	8	3	7	5	
Episode length	41	38	38	33	20	50	43	

Table 1: The task configuration of the tasks in *SymWoB* domain. Each task is parameterized by different subtask graphs, and the episode length is manually set according to the challengeness of the tasks.

In this paper, we introduce the *SymWoB* domain, which is a challenging symbolic environment that aims to reflect the hierarchical and compositional aspects of the checkout processes in the real-world websites. There are total 15 different *SymWoB* websites that are symbolic implementations of the actual websites: **Amazon**, **Apple**, **BestBuy**, **Dick's**, **eBay**, **Expedia**, **Ikea**, **Lego**, **Lenox**, **Omahasteaks**, **Swarovski**, **Thriftbooks**, **Todaytix**, **Walgreens**, and **Walmart**. All of these websites are generated by analyzing the corresponding real websites and reflecting their key aspects of checkout process. The main goal of each website is to navigate through the web pages within the website by clicking and filling in the web elements with proper information which leads to the final web page that allows the agent to click on the `Place_Order` button, which indicates that the agent has successfully finished the task of checking out.

7.1 IMPLEMENTATION DETAIL

In this section, we describe the detailed process of implementing an existing website into a symbolic version. We first fill out the shopping cart with random products, and we proceed until placing the order on the actual website. During the process, we extract all the interactable web elements on the webpage. We repeat this for all the websites and form a shared subtask pool where similar web elements in different websites that have same functionality are mapped to the same subtask in the shared subtask pool. Then, we extract the precondition relationship between subtasks from the website and form the edges in the subtask graph accordingly. Finally, we implement the termination condition and the subtask reward to the failure distractor (See Section 7.3) and the goal subtasks.

7.2 COMPARISON OF THE WEBSITES

The agent's goal on every website is the same, that is placing the checkout order. However, the underlying subtask graphs, or task structure, of the websites are quite diverse, making the task much more challenging for the agent. Figures 3-17 visualize the ground truth subtask graph of all the websites. One of the major sources of diversity in subtask graphs is in the various ordering of the web pages. In a typical website's checkout process, some of the most common web pages include the shipping, billing, and payment web pages, each of which has a collection of corresponding subtasks. In Figure 3, for example, the shipping web page is represented by the collection of the subtasks on the left side from `Fill_Zip` to `Fill_Last` and `Click_ContinueShipping`, and these come *before* the payment web page that is represented by the subtasks on the right side from `Click_Credit` to `Click_ContinuePayment`. On the other hand, in Figure 7, the similar web pages are either connected in a different ordering, from payment to shipping web page, or placed on the same line side by side. Since the web pages can vary on how they are ordered, it allows the subtask graphs to have a variety of shapes such as deep and narrow as in Figure 9 or wide and shallow as in Figure 7. Different shape of the subtask graphs means different precondition between the tasks, making it non-trivial for the agent to transfer its knowledge about one to the other.

Another major source of diversity is the number of web elements in each web page. Let's compare the web elements of the shipping web page in Figure 4 and Figure 5. These are the subtasks that are connected to `Click_ContinueShipping` and as well as itself. We can see that the two websites do not have the same number of the web elements for the shipping web pages: the **BestBuy** website requires more shipping information to be filled out than the **Dick's** website. Such variety in the number of web elements, or subtasks, allows the subtask graphs of the websites to have diverse preconditions as well.

7.3 DISTRACTOR SUBTASKS

In addition to the different task structures among the websites, there are also *distractor* subtasks in the websites that introduces challenging components of navigating the real-world websites. There are two different types of distractor subtasks: the one that terminates the current episode with a negative reward and the another one that has no effect. The former, which we also call it the *failure* distractor subtask, represents the web elements that lead the agent to some external web pages like `Help` or `Terms_of_Use` button. The latter is just called the distractor subtask, where executing the subtask does not contribute to progressing toward the goal (*e.g.*, `Fill Coupon` subtask in **BestBuy**). Each website has varying number of distractor subtasks and along with the shallowness of the task structure, the number of distractor subtasks significantly affects the difficulty of the task.

8 DETAILS OF THE AGENT IMPLEMENTATION

We implement all of our algorithms, including both MTSGI and the baselines, on top of the recently introduced RL framework called Acme [Hoffman et al., 2020].

8.1 MSGI

Similar to the MTSGI agent, the MSGI agent uses the soft-version of UCB exploration policy as the adaptation policy, instead of its original hard-version due to a better performance. Furthermore, MSGI also uses inductive logic programming (ILP) in order to infer the subtask graph of the current task. But unlike the MTSGI agent that learns prior across the tasks through multi-task learning, the MSGI agent does not exploit the subtask graphs inferred from the previous tasks.

8.2 HRL

The hierarchical RL (HRL) agent is an option-based agent that can execute temporally extended actions. For the *Mining* domain, where there is a spatial component in the observation input, we use convolutional neural network (CNN) in order to encode the spatial information and get concatenated along with the other additional inputs, which are encoded using fully-connected (FC) networks. The concatenated embedding then gets passed on to GRU, which is followed by two separate heads for value and policy function outputs.

The details of the HRL architecture for *Mining* domain are: Conv1(16x1x1-1)-Conv2-(32x3x3-1)-Conv3(64x3x3-1)-Conv4(32x3x3-1)-Flatten-FC(256)-GRU(512). The HRL architecture for *SymWoB* domain is almost identical except it replaces the CNN module with fully-connected layers for processing non-spatial information: FC(64)-FC(64)-FC(50)-FC(50)-GRU(512). We use ReLU activation function in all the layers.

For training, we use multi-step actor-critic (A2C) in order to train HRL. We use multi-step learning of $n = 10$, learning rate 0.002, entropy loss weight of 0.01 and critic loss weight of 0.5. We use RMSProp optimizer to train the networks, where its decay rate is set to 0.99 and epsilon to 0.00001. We also clip the gradient by setting the norm equal to 1.0. Most importantly, the network parameters of the HRL agent gets reset for every new task since HRL is not a meta-RL agent.

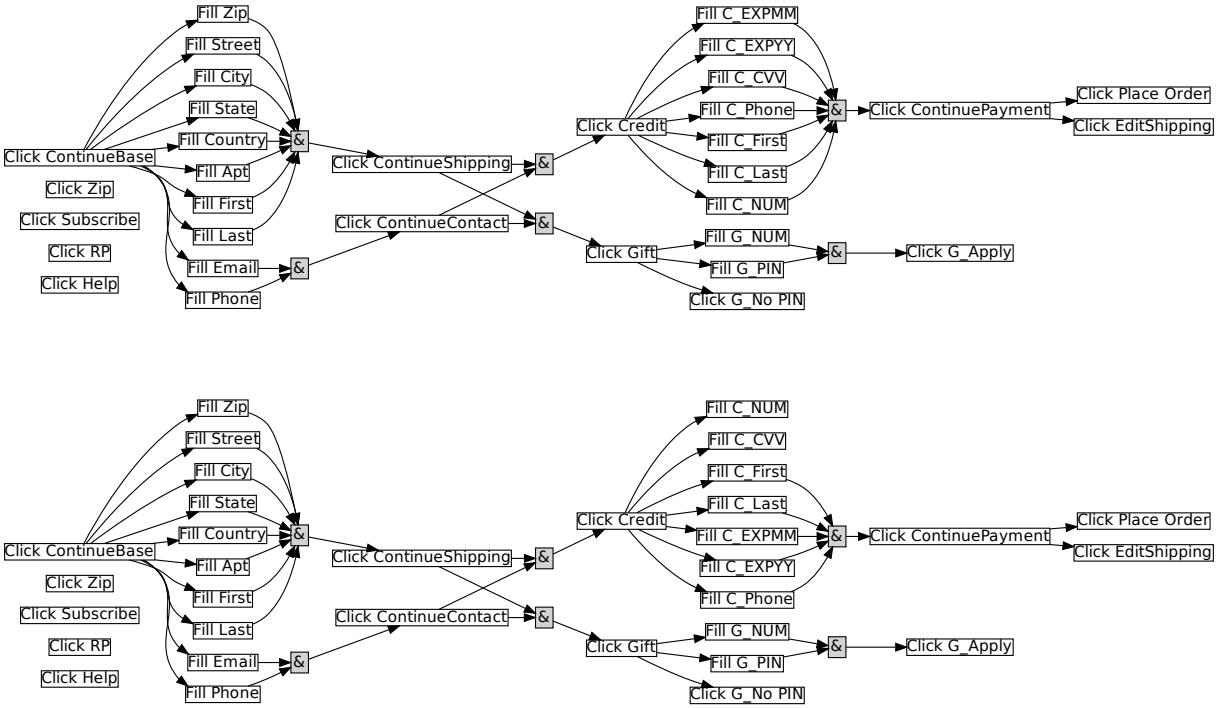

Figure 3: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **Walmart** website.

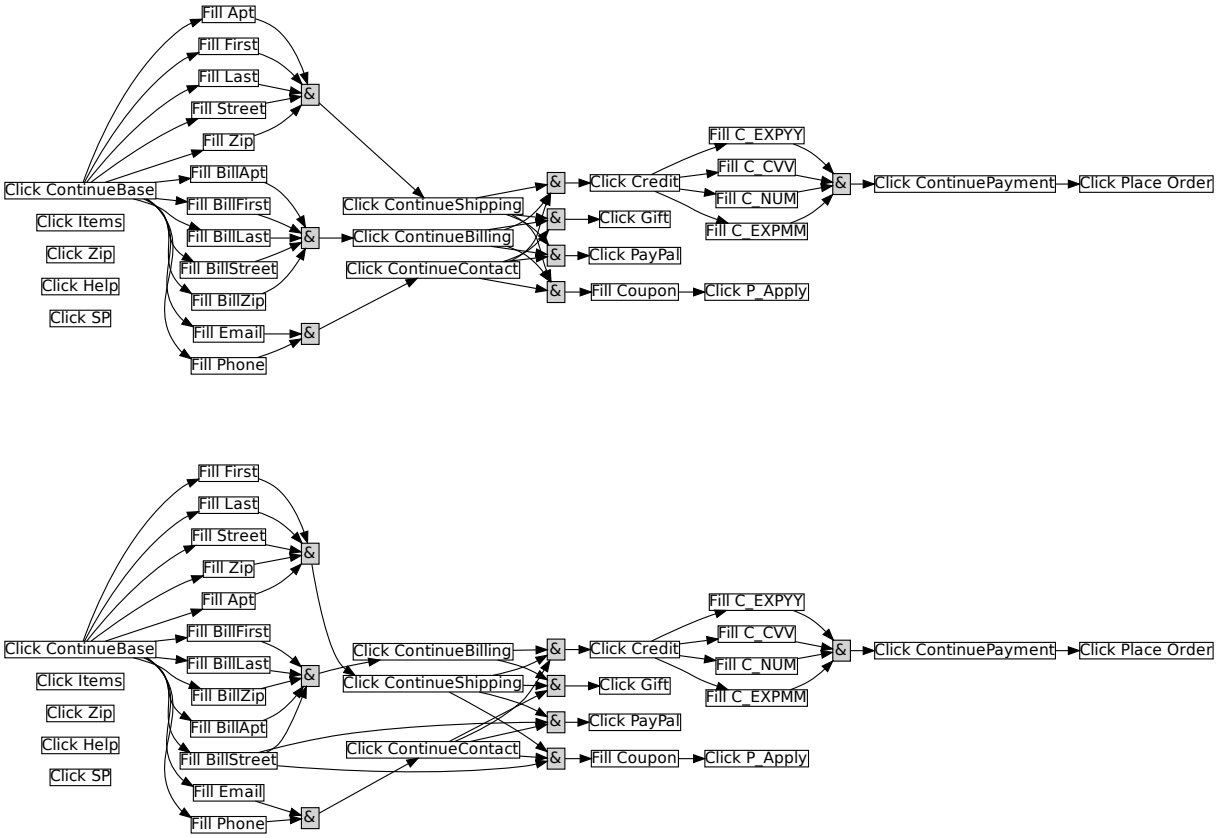

Figure 4: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **Dick's** website.

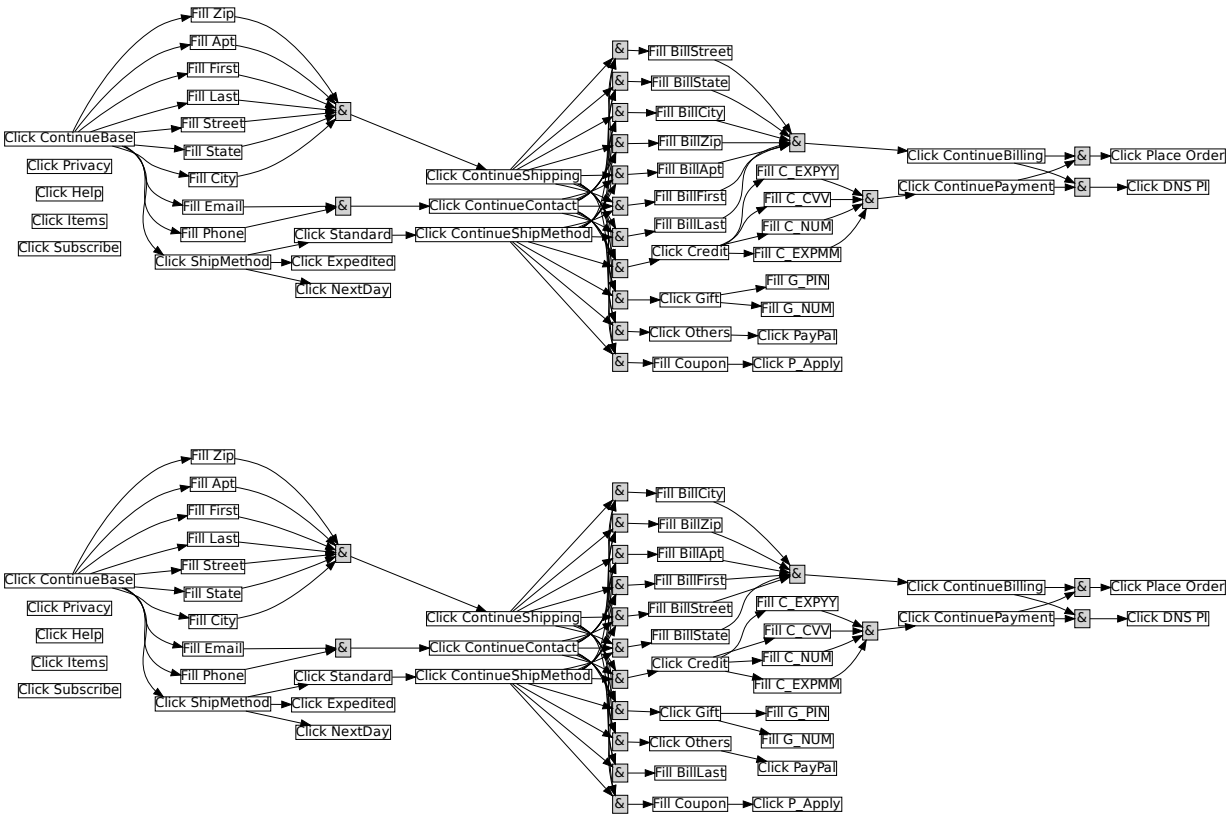

Figure 5: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **BestBuy** website.

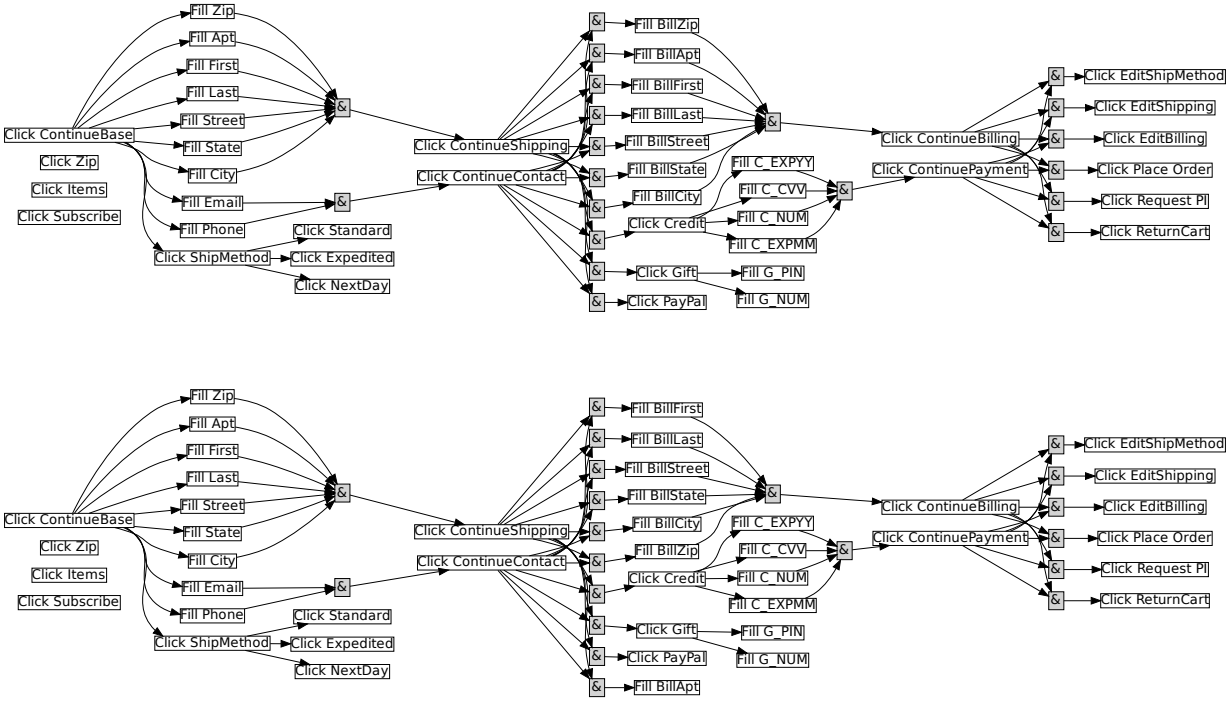

Figure 6: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **Apple** website.

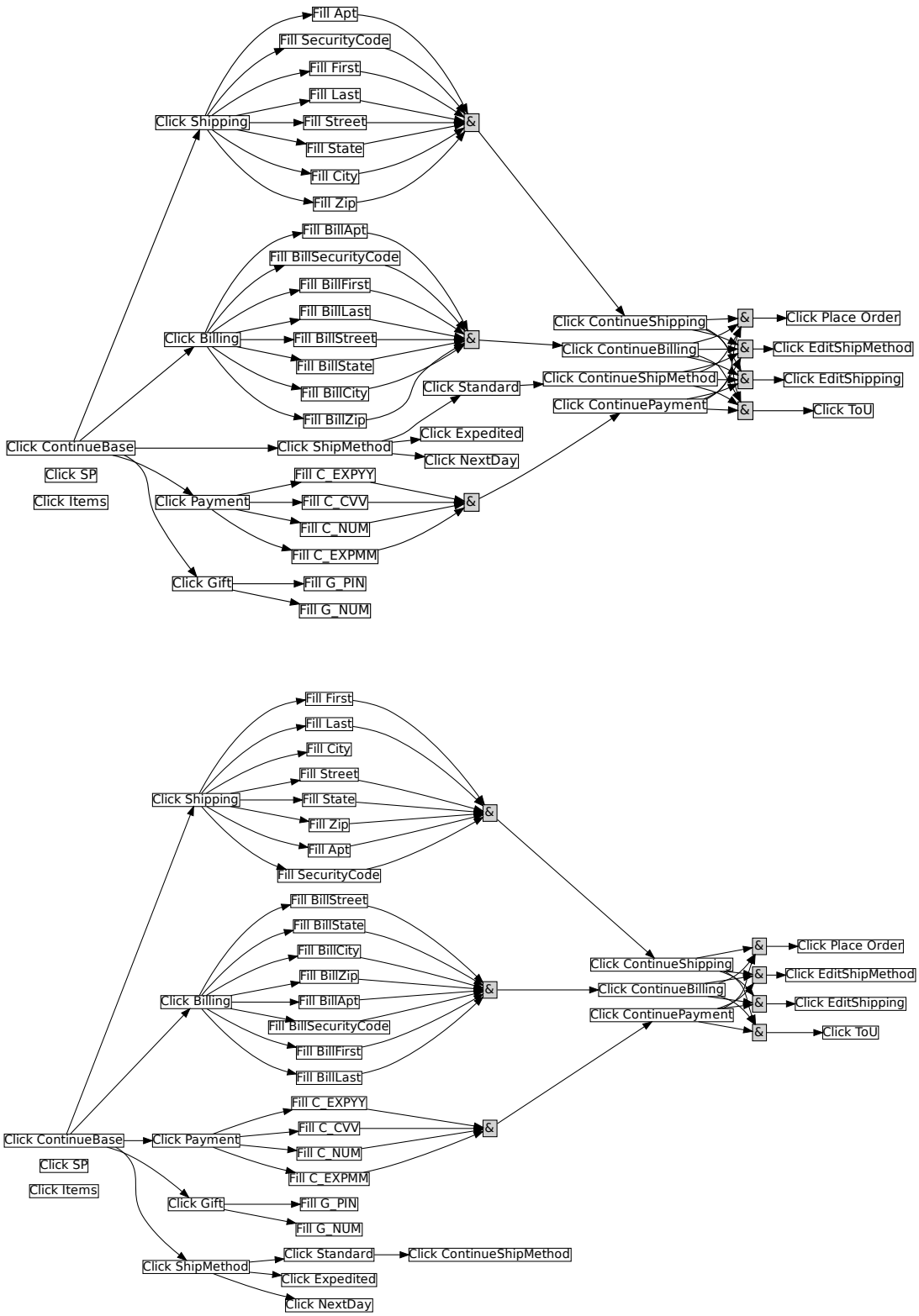

Figure 7: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **Amazon** website.

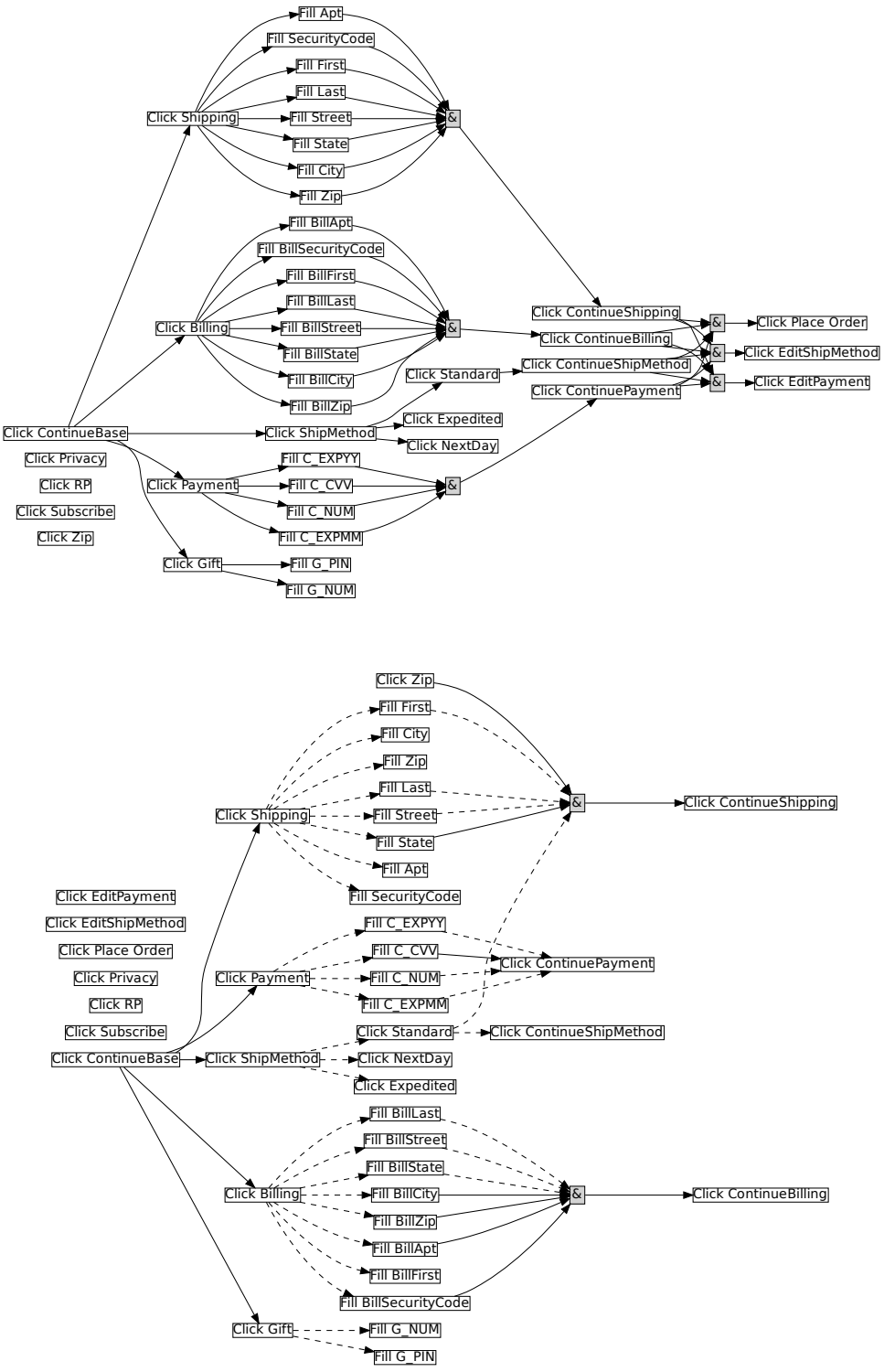

Figure 8: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **eBay** website.

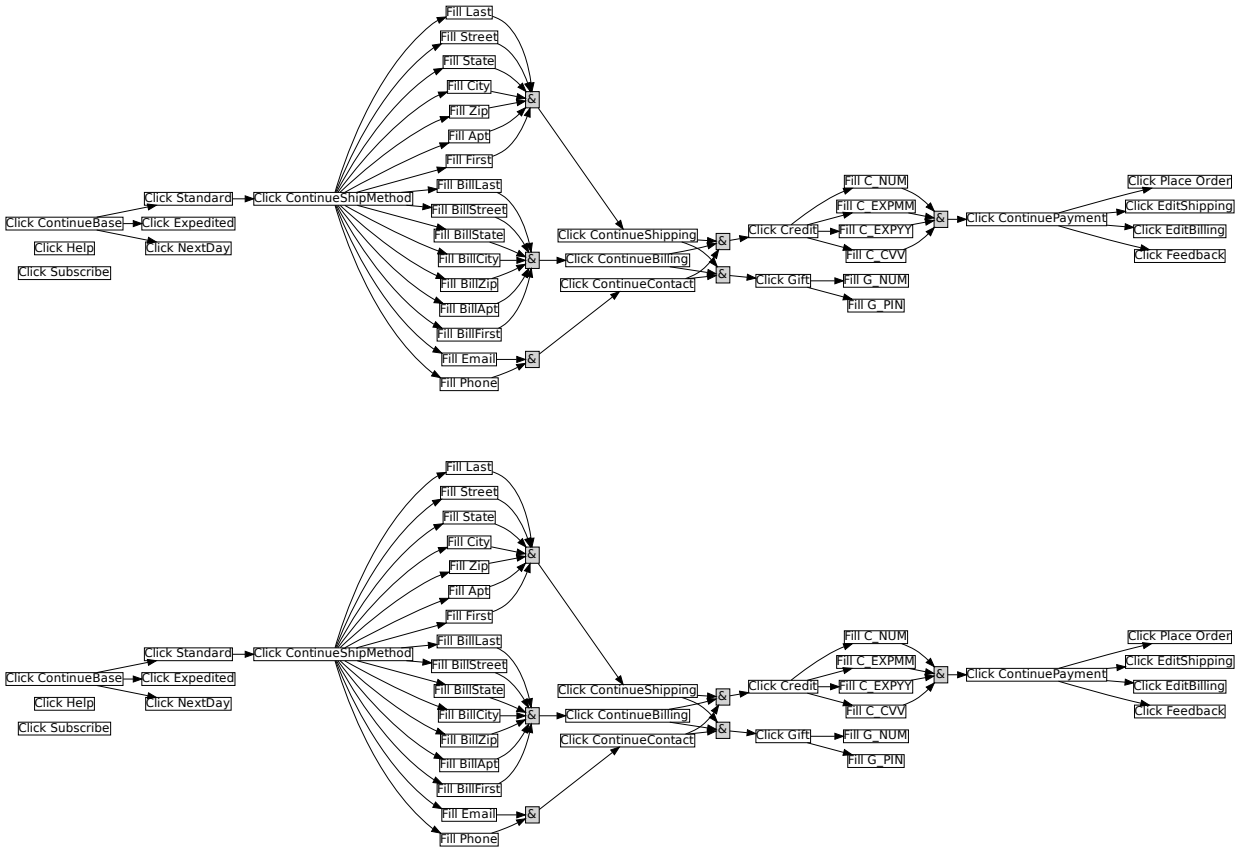

Figure 9: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **Ikea** website.

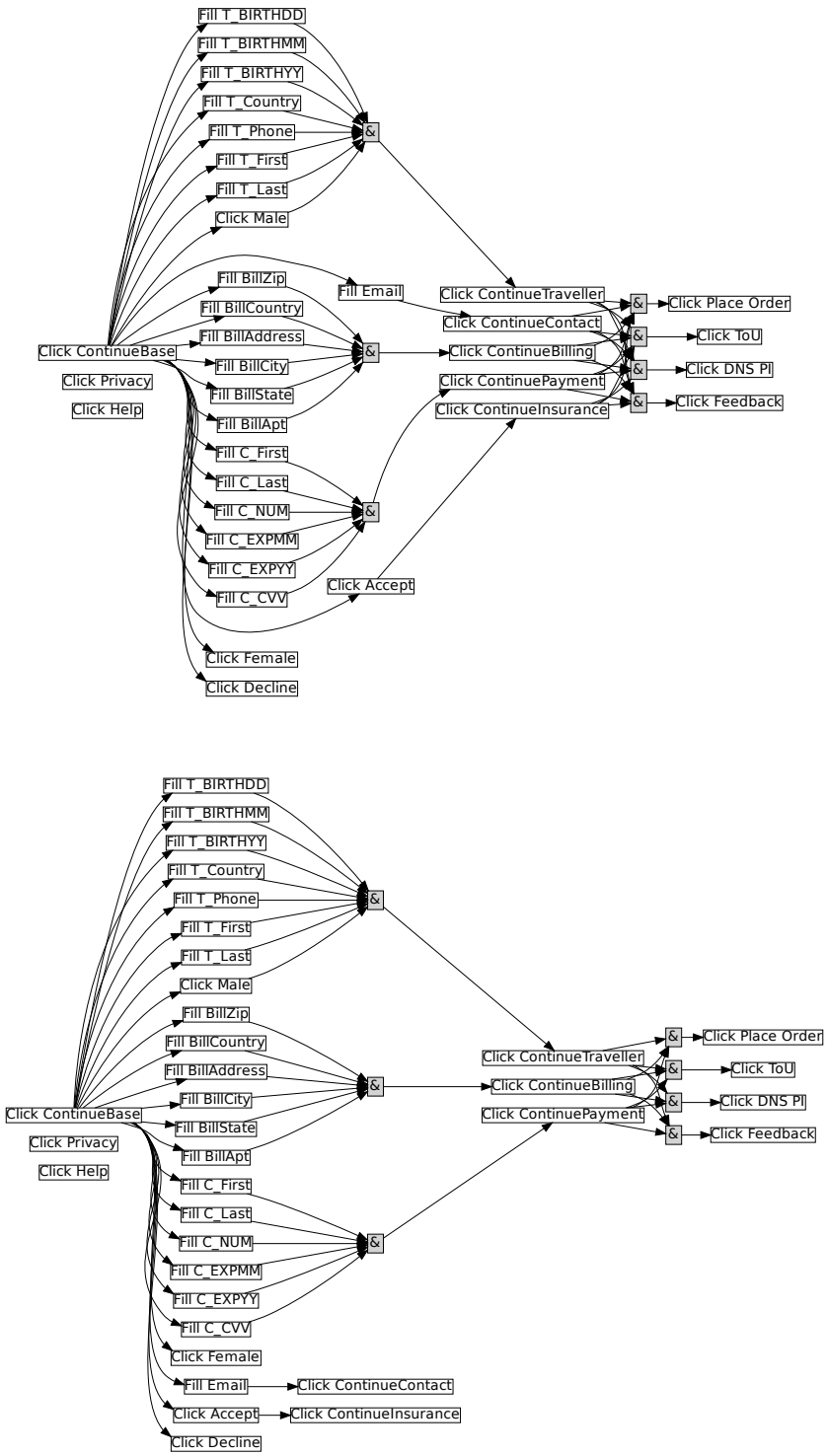

Figure 10: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **Expedia** domain.

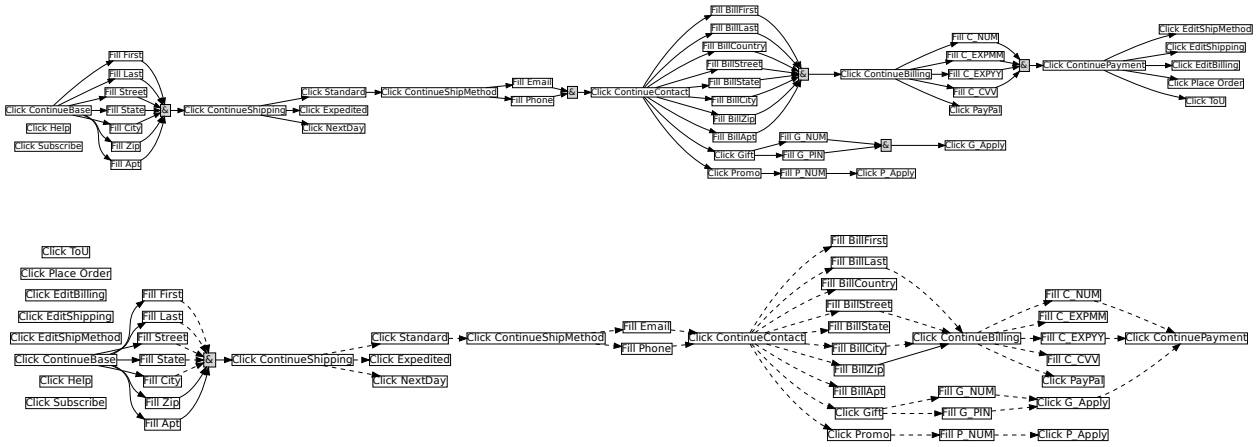

Figure 11: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **Lego** domain.

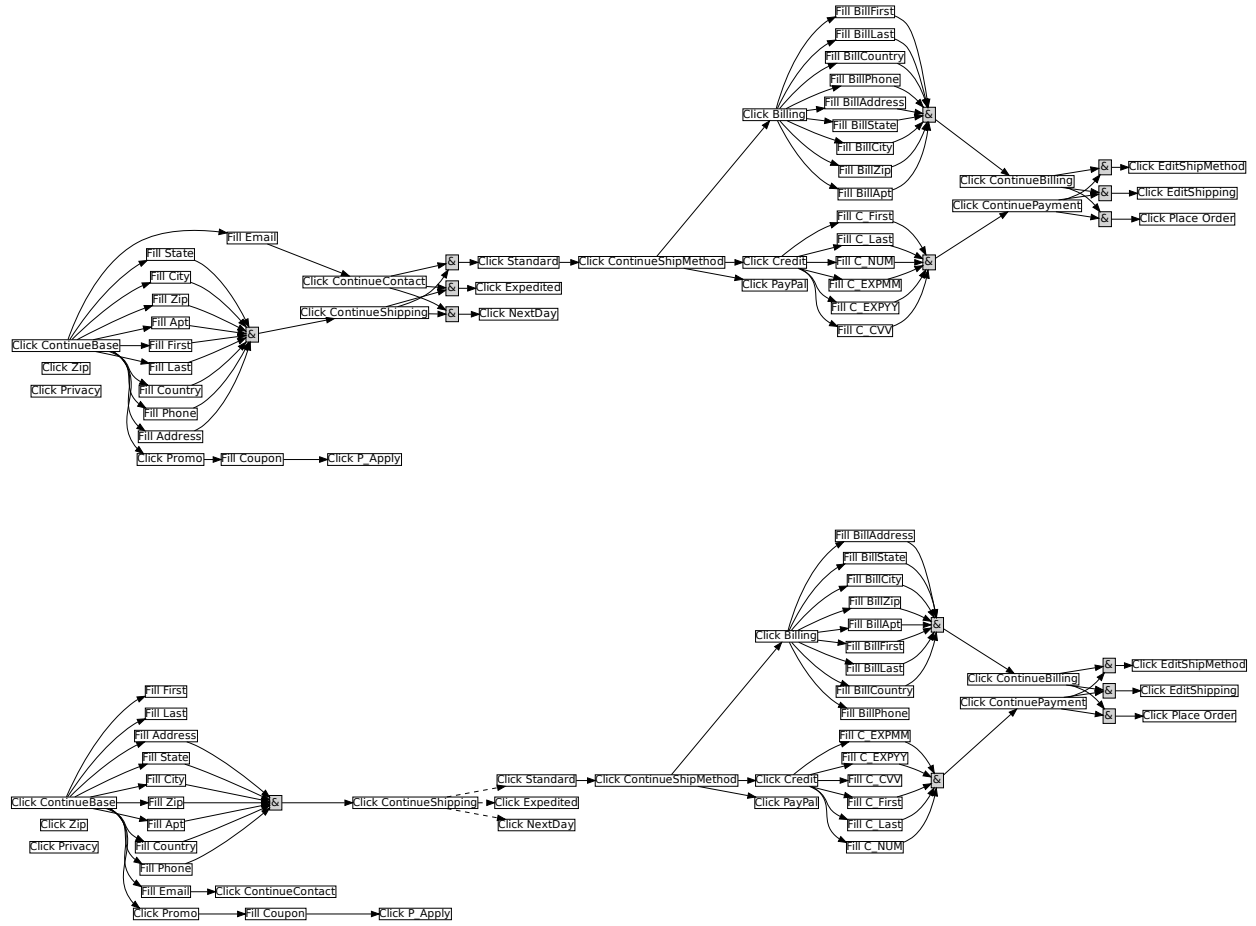

Figure 12: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **Lenox** domain.

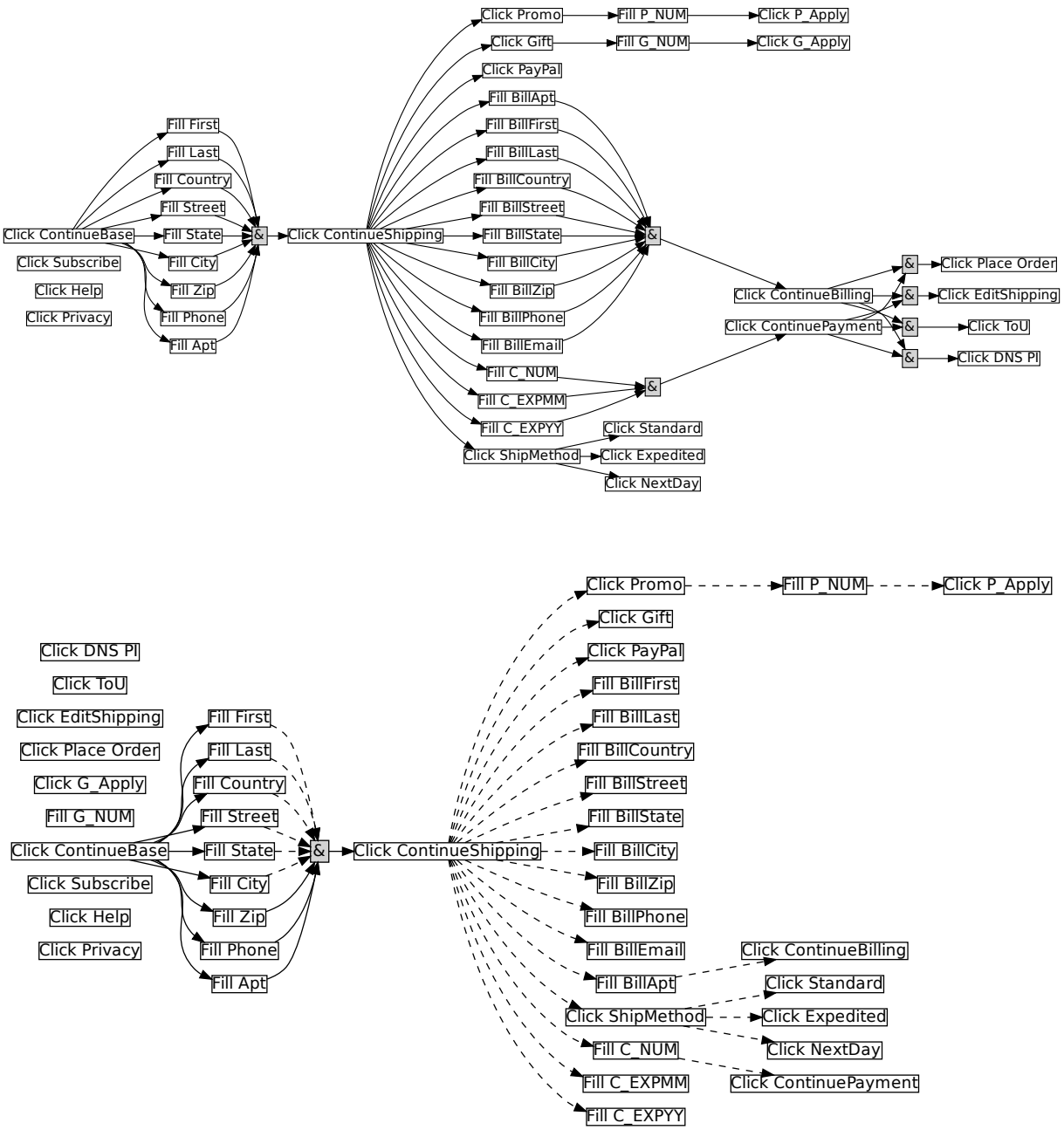

Figure 13: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **Omahasteaks** domain.

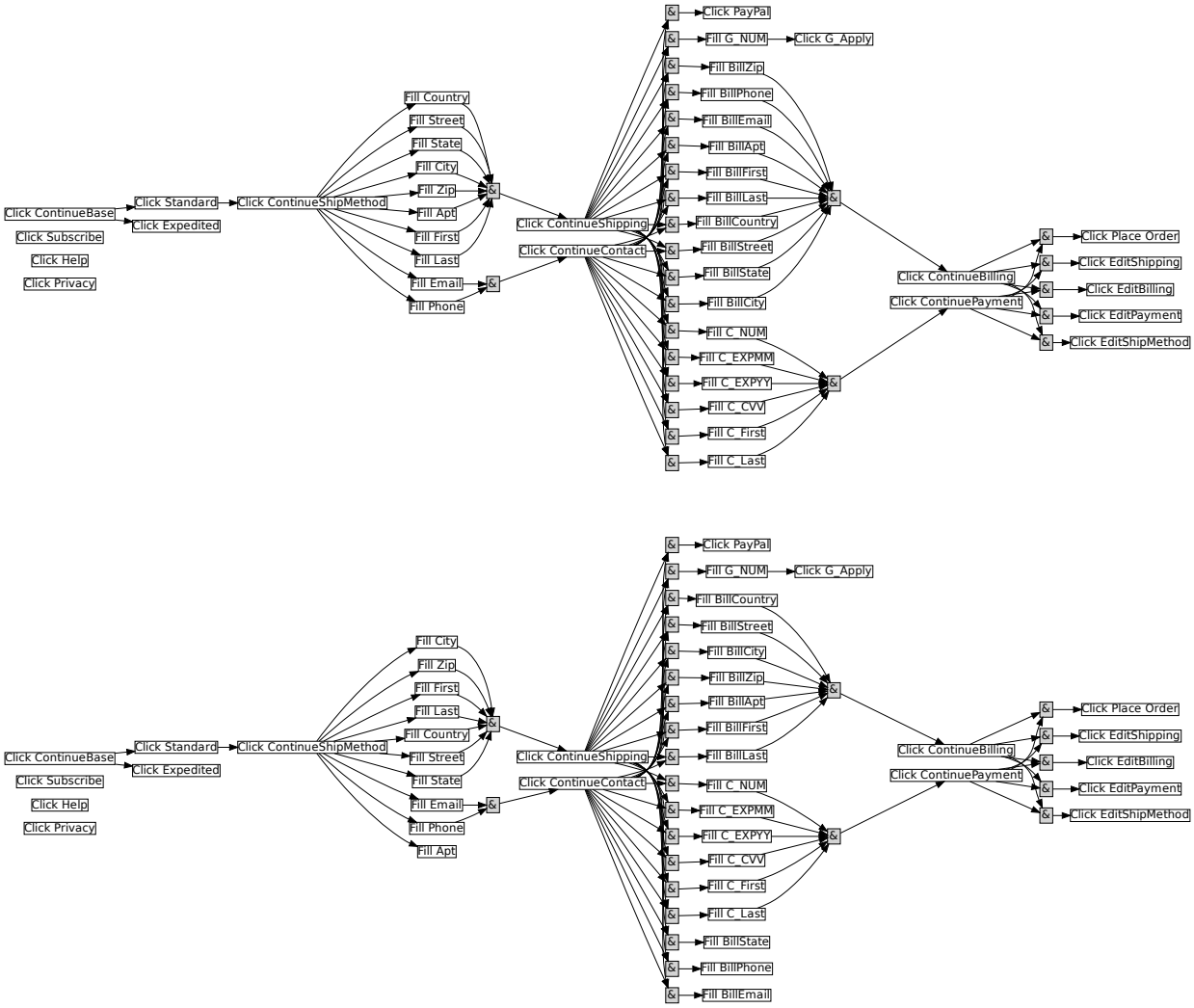

Figure 14: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **Swarovski** domain.

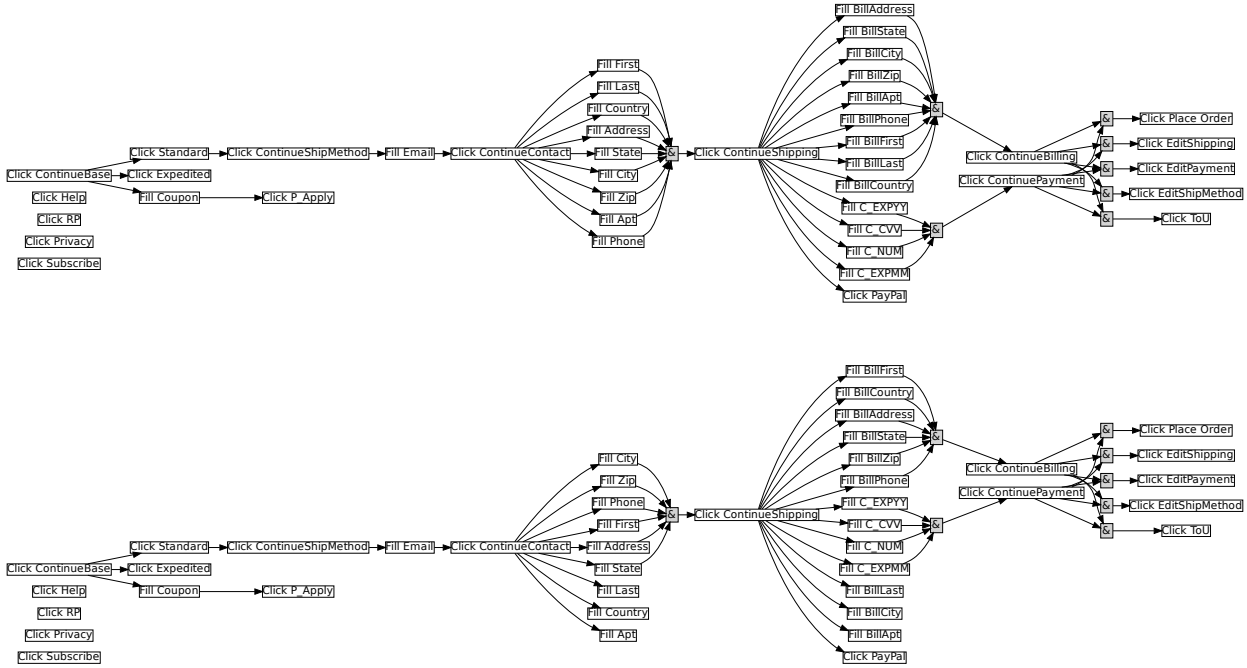

Figure 15: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **Thriftbooks** domain.

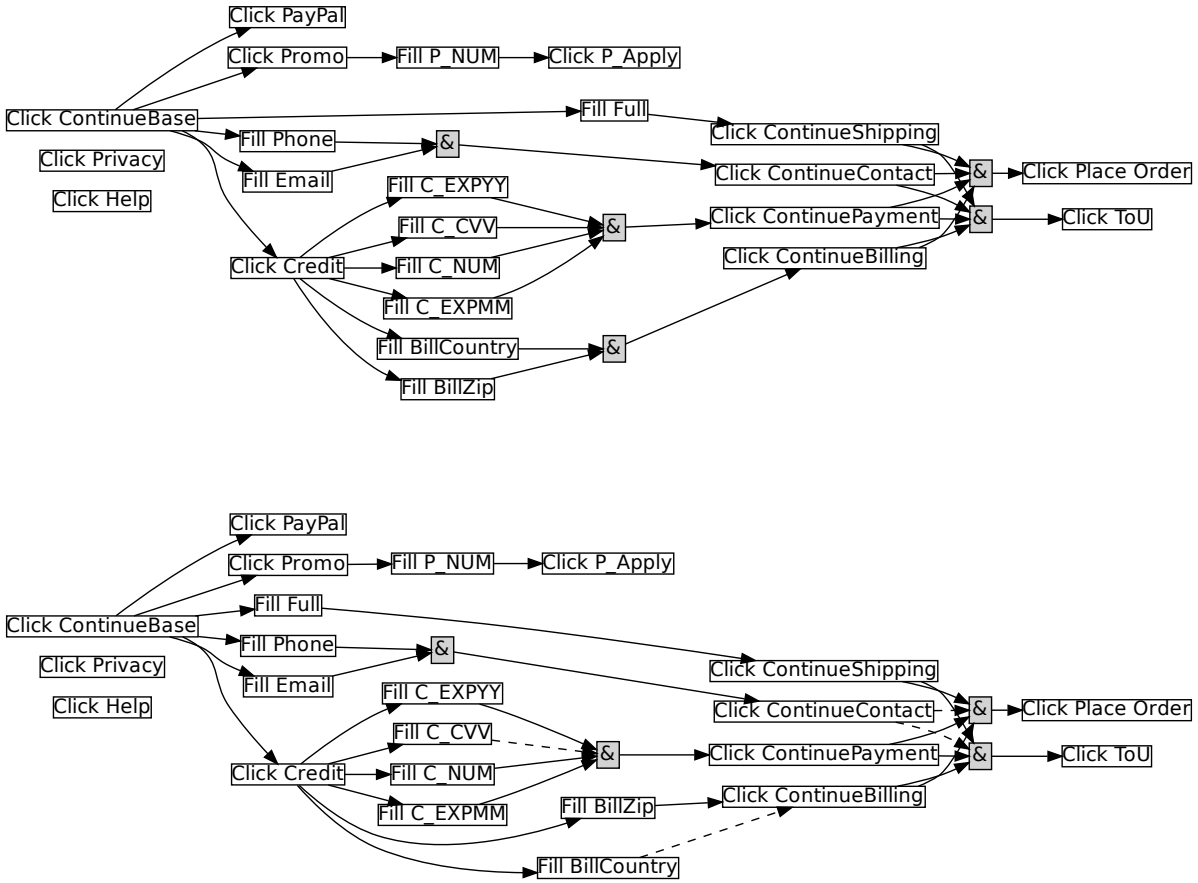

Figure 16: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **Todaytix** domain.

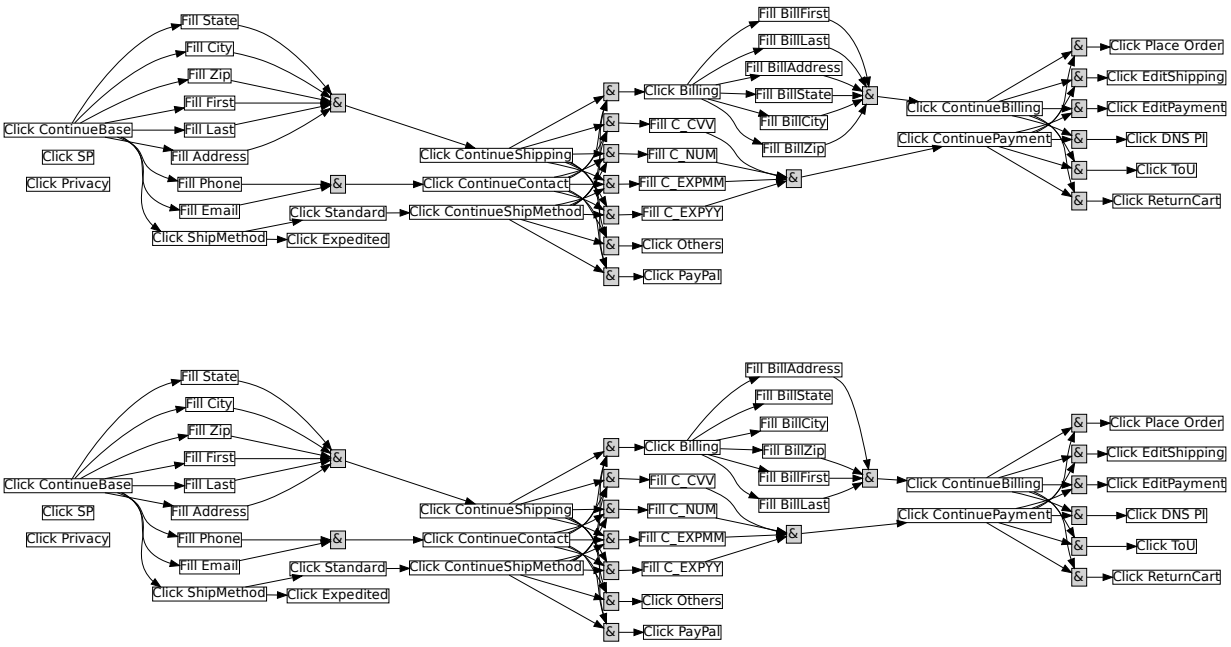

Figure 17: *(Top)* The ground-truth and *(Bottom)* the inferred subtask graphs of **Walgreens** domain.

References

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *ICML*, 2017.

Leo Breiman. *Classification and regression trees*. Routledge, 1984.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pages 794–803. PMLR, 2018.

Behzad Ghazanfari and Matthew E Taylor. Autonomous extracting a hierarchical structure of tasks in reinforcement learning and multi-task reinforcement learning. *arXiv preprint arXiv:1709.04579*, 2017.

Izzeddin Gur, Ulrich Rueckert, Aleksandra Faust, and Dilek Hakkani-Tur. Learning to navigate the web. *arXiv preprint arXiv:1812.09195*, 2018.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

Matt Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, Sarah Henderson, Alex Novikov, Sergio Gómez Colmenarejo, Serkan Cabi, Caglar Gulcehre, Tom Le Paine, Andrew Cowie, Ziyu Wang, Bilal Piot, and Nando de Freitas. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020. URL `https://arxiv.org/abs/2006.00979`.

De-An Huang, Suraj Nair, Danfei Xu, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese, and Juan Carlos Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. *arXiv preprint arXiv:1807.03480*, 2018.

Sheng Jia, Jamie Ryan Kiros, and Jimmy Ba. DOM-q-NET: Grounded RL on structured language. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HJgd1nAqFX`.

George Konidaris and Andrew Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 489–496, 2006.

Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer, 2012.

Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 544–551, 2008.

Xingyu Lin, Harjatin Singh Baweja, George Kantor, and David Held. Adaptive auxiliary task weighting for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Changsong Liu, Shaohua Yang, Sari Iaba-Sadiya, Nishant Shukla, Yunzhong He, Song-chun Zhu, and Joyce Chai. Jointly learning grounded task structures from language instruction and visual demonstration. In *EMNLP*, 2016.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. *arXiv preprint arXiv:1802.08802*, 2018.

Stephen Muggleton. Inductive logic programming. *New Gen. Comput.*, 8(4):295–318, February 1991. ISSN 0288-3635. doi: 10.1007/BF03037089. URL `http://dx.doi.org/10.1007/BF03037089`.

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *ICML*, 2017.

Lerrel Pinto and Abhinav Gupta. Learning to push by grasping: Using multiple tasks for effective learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2161–2168. IEEE, 2017.

Earl D Sacerdoti. The nonlinear nature of plans. Technical report, STANFORD RESEARCH INST MENLO PARK CA, 1975a.

Earl D Sacerdoti. A structure for plans and behavior. Technical report, SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1975b.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pages 3135–3144, 2017.

Sungryull Sohn, Junhyuk Oh, and Honglak Lee. Hierarchical reinforcement learning for zero-shot generalization with subtask dependencies. In *NeurIPS*, pages 7156–7166, 2018.

Sungryull Sohn, Hyunjae Woo, Jongwook Choi, and Honglak Lee. Meta reinforcement learning with autonomous inference of subtask dependencies. In *International Conference on Learning Representations*, 2019.

Austin Tate. Generating project networks. In *Proceedings of the 5th international joint conference on Artificial intelligence-Volume 2*, pages 888–893. Morgan Kaufmann Publishers Inc., 1977.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pages 1015–1022, 2007.

Yu Zhang and Dit-Yan Yeung. A regularization approach to learning task relationships in multitask learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(3):1–31, 2014.