Appendix

Hypergraph Construction

Algorithm 1 presents the pseudo-code for constructing a reduced hypergraph, given a training example $\boldsymbol{\mathcal{D}}$, a set of mode declarations and a set of template predicates. Constants in $\boldsymbol{\mathcal{D}}$ are appearing as nodes and true ground atoms as hyperedges, connecting the nodes appearing as their arguments. The hypergraph is constructed by only allowing input and output nodes, because only the search space defined by the mode declarations will be eventually searched. Note that template predicates are not added in the hypergraph because they are not allowed to appear in the body of a definite clause.

Algorithm 1 ConstructHG($\boldsymbol{\mathcal{D}}$, modes, $\mathcal{P}$)

```
Input: $\boldsymbol{\mathcal{D}}$: training example, modes: mode declarations, $\mathcal{P}$: set of template predicates
Output: $HG$: hypergraph
1: for all constant $k \in \boldsymbol{\mathcal{D}}$ do
2:     $HG[k] = \emptyset$
3: for all true ground atom $\mathrm{p}(k_1, \ldots, k_r) \in \boldsymbol{\mathcal{D}}$ & $\mathrm{p} \notin \mathcal{P}$ do
4:     for all constant $k_i \in \{k_1, \ldots, k_r\}$ do
5:         if isInputOrOutputVar($k_i$, modes) then $HG[k_i] = HG[k_i] \cup \{\mathrm{p}(k_1, \ldots, k_r)\}$
   return $HG$
```

Algorithm 2 InitialSet(q, $\boldsymbol{\mathcal{D}}$, $\mathbf{T}$)

```
Input: q: incorrectly predicted atom, $\boldsymbol{\mathcal{D}}$: training example, $\mathbf{T}$: path template
Output: $\mathcal{I}$: a set of grounded template predicates
1: $\mathcal{I} = \emptyset$
2: for all axiom $\alpha \in \mathbf{T}$ do
3:     if $\exists$ literal $\ell \in \alpha$ : signature($\ell$)=signature($q$) & isPositive($\ell$)=isTrue($q, \boldsymbol{\mathcal{D}}$) then
4:         $\theta$-substitute $\beta = \alpha\theta$ where $\theta = \{\mathrm{variables}(\ell) \to \mathrm{constants}(q)\}$
5:         $\tau$ = templateAtom($\beta$)
6:         if $\exists$ variable $v \in \tau$ then
7:             $\mathcal{L} = \emptyset$
8:             for all literal $\ell \in \beta : \exists\ v \in \tau \wedge v \in \ell$, do
9:                 $\mathcal{L} = \mathcal{L} \cup \ell$
10:            for all literal $\ell \in \mathcal{L}$ do
11:                $\mathbf{K}_\ell = \{k_1, \ldots, k_n\} \in \boldsymbol{\mathcal{D}} : \ell(k_1, \ldots, k_n) \Rightarrow \top$
12:            for all row $r = \{k_1, \ldots, k_n\} \in \bowtie_{\ell=1}^{|\mathcal{L}|} \mathbf{K}_\ell$ do
13:                $\theta$-substitute $\gamma = \beta\theta$ where $\theta = \{\mathit{variables}(\beta) \Rightarrow r\}$
14:                $\mathcal{I} = \mathcal{I} \cup$ templateAtom($\gamma$)
15:        else
16:            $\mathcal{I} = \mathcal{I} \cup \tau$
17: return $\mathcal{I}$
```

Initial Search Set

Algorithm 2 presents the pseudo-code for extracting ground template predicates (initial search set $\mathcal{I}$), used for relational pathfinding, given a training example $\boldsymbol{\mathcal{D}}$, a path template $\mathbf{T}$ and an incorrectly predicted atom q. The algorithm considers each axiom $\alpha \in \mathbf{T}$ in turn and checks if α contains a predicate having identical atom signature and literal type with q. Atom signature is the combination of predicate symbol and arity. The literal type can be either positive or negative. For each axiom satisfying these properties, it substitutes the constants of q into α and checks whether the template predicate still contains any variables. If there are no variables left, it adds the grounded template predicate to $\mathcal{I}$ and moves to the next axiom. Otherwise, it searches for all literals in the axiom sharing the remaining variables with the template predicate. For these literals, it then searches the training data $\boldsymbol{\mathcal{D}}$ for all jointly ground instantiations among those satisfying the axiom. For each of them, it substitutes the remaining variables of the template predicate and adds the resulted ground template predicate to $\mathcal{I}$.

Mode-Guided Search

Algorithm 3 presents the pseudo-code for mode-guided relational pathfinding that searches the hypergraph for appropriate bodies explaining the grounded template predicates of the initial search set $\mathcal{I}$. Given a starting path having only a single grounded template predicate it recursively adds to the path hyperedges (i.e., ground atoms) that satisfy the mode declarations. The search terminates when the path reaches a specified maximum length or when no new hyperedges can be added. The algorithm stores all paths encountered during the search.

Algorithm 3 ModeGuidedSearch(curPath, $\mathcal{V}$, HG, mode, maxLength, paths)

```
Output: paths: a set of paths
 1: if |curPath| < maxLength then
 2:     for all constant $k \in \mathcal{V}$ do
 3:         for all p$(k_1, \ldots, k_r) \in HG[k]$ do
 4:             if canAdd(p, curPath, mode) then
 5:                 if curPath $\notin$ paths then
 6:                     curPath = curPath $\cup \{$p$(k_1, \ldots, k_r)\}$
 7:                     paths = paths $\cup \{p(k_1, \ldots, k_r)\}$
 8:                     $\mathcal{V}' = \emptyset$
 9:                     for all $k_i \in \{k_1, \ldots, k_r\}$ do
10:                         if $k_i \notin \mathcal{V}$ & isInputOrOutputVar($k_i$, mode) then
11:                             $\mathcal{V} = \mathcal{V} \cup \{k_i\}$
12:                             $\mathcal{V}' = \mathcal{V}' \cup \{k_i\}$
13:                     ModeGuidedFindPath(curPath,$\mathcal{V}$,$HG$,mode,maxLength,paths)
14:                     curPath = curPath $\setminus$ p
15:                     $\mathcal{V} = \mathcal{V} \setminus \mathcal{V}'$
```

Online Structure Learning using Background Knowledge Axiomatization

Algorithm 4 presents the complete `OSL`α procedure.

Algorithm 4 OSLa($\mathcal{BK}$,$\mathcal{P}$,modes,maxLength,μ,λ,η,δ)
Input: $\mathcal{BK}$: background knowledge, $\mathcal{P}$: template predicates
modes: mode declarations, maxLength: max. number of hyperedges in a path
μ: evaluation threshold, λ, η, δ: AdaGrad parameters

```
1: Partition $\mathcal{BK}$ into $\mathcal{A}$ and $\mathcal{B}$ and create templates $\mathbf{T}$ from $\mathcal{A}$
2: Initialize resulting theory $\mathcal{R}_0 = \mathcal{B}$ and weight vector $\mathbf{w}_0$ = initialValue
3: for $t = 1$ to $T$ do
4:     Receive a training example $\boldsymbol{\mathcal{D}}_t$=$(\mathbf{x}_t, \mathbf{y}_t)$
5:     Predict $\mathbf{y}_t^P = \text{argmax}_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}}\langle \mathbf{w}, \mathbf{n}(\mathbf{x}_t, \mathbf{y})\rangle$ and compute $\Delta y_t = \mathbf{y}_t \setminus \mathbf{y}_t^P$
6:     if $\Delta y_t \neq \emptyset$ then
7:         $HG$ = ConstructHG$\big((\mathbf{x}_t, \mathbf{y}_t), \text{modes}, \mathcal{P}\big)$      ▷ Reduced hypegraph construction
8:         paths= $\emptyset$
9:         for all incorrectly predicted query atom q $\in \Delta y_t$ do
10:            for all template $T_i \in \mathbf{T}$ do
11:                $\mathcal{I}$ = InitialSet$\big(\text{q}, (\mathbf{x}_t, \mathbf{y}_t), T_i\big)$
12:                $\mathcal{V} = \emptyset$                                   ▷ A set of constants
13:                for all $\tau(k_1, \dots, k_n) \in \mathcal{I}$ do
14:                    for all $k_i \in \{k_1, \dots, k_n\}$ do
15:                        if isInputOrOutputVar($k_i$, modes) then
16:                            $\mathcal{V} = \mathcal{V} \cup k_i$
17:                    ModeGuidedSearch$\big(\text{q}, \mathcal{V}, HG, \text{modes}, \text{maxLength}, \text{paths}\big)$
18:    $\mathcal{DC}_t$ = CreateDefinitions($\mathcal{DC}_{t-1}$,paths,modes)
19:    $\mathcal{C}_t$ = CreateClauses($\mathcal{DC}_t$, modes)
20:    for $i = 1$ to $|\mathcal{C}_t|$ do
21:        if $\Delta\mathbf{n}_{\mathcal{C}_{t,i}} \leq \mu$ then Remove def. clause $dc_i$ corresponding to $\mathcal{C}_{t,i}$ from $\mathcal{DC}_t$
22:    $\mathcal{R}_t = \mathcal{B} \cup$ CreateClauses($\mathcal{DC}_t$, modes)
23:    for $i = 1$ to $|\mathcal{R}_t|$ do
24:        if $\exists\, c_{i,t-1} \in \mathcal{R}_{t-1}$, $c_{i,t} \in \mathcal{R}_t$ : $c_{i,t-1}$ $\theta$-subsumes $c_{i,t}$ then $w_{i,t}$=$w_{i,t-1}$
25:        else $w_{i,t}$=initialValue
26:    AdaGrad($\mathbf{w}_t, \Delta\mathbf{n}_{\mathcal{R}_t}, \lambda, \eta, \delta$)
```