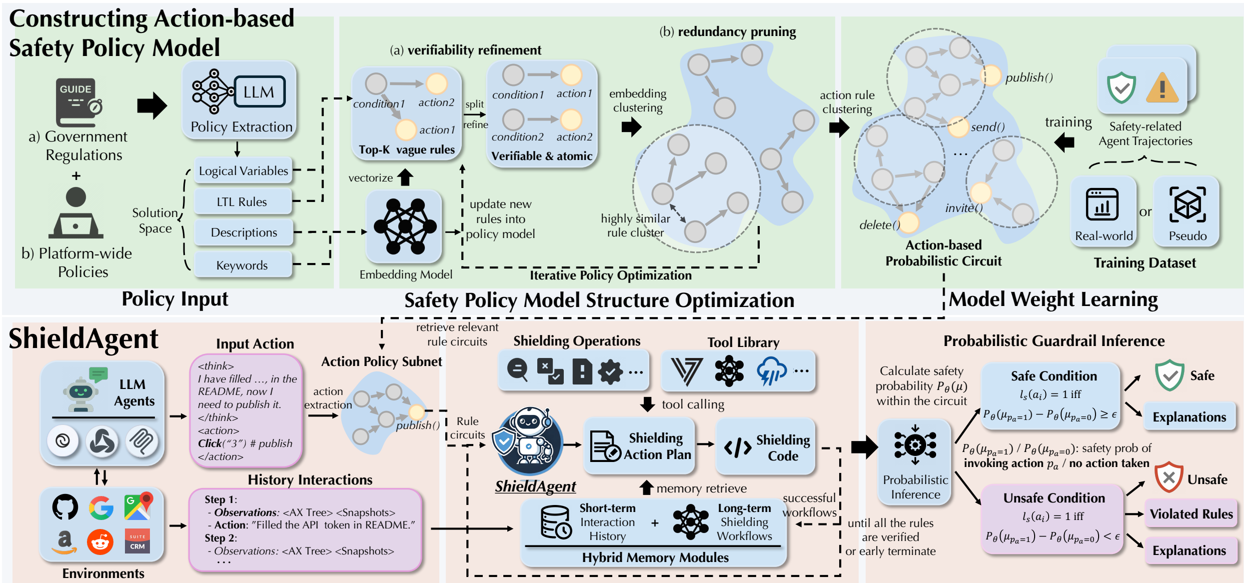

Figure-r. 1. **Overview of ShieldAgent. (Top)** From AI regulations (e.g. EU AI Act) and platform-specific safety policies, ShieldAgent first extracts verifiable rules and iteratively refines them to ensure each rule is accurate, concrete, and atomic. It then clusters these rules and assembles them into an action-based safety policy model, associating actions with their corresponding constraints (with weights learned from real or simulated data). **(Bottom)** During inference, ShieldAgent retrieves relevant rule circuits w.r.t. the invoked action and performs action verification. By referencing existing workflows from a hybrid memory module, it first generates a step-by-step shielding plan with operations supported by a comprehensive tool library to assign truth values for all predicates, then produces executable code to perform formal verification for actions. Finally, it runs probabilistic inference in the rule circuits to provide a safety label and explanation and reports violated rules.

Algorithm 1 ShieldAgent Inference Procedure

Require: Interaction history $\mathcal{H}_{<i} = \{(o_j, a_j) \mid j \in [1, i-1]\}$ from the target agent; Current observation o_i; Agent output a_i; Safety policy model $\mathcal{G}_{\text{ASPM}} = \big(\mathcal{P}, \mathcal{R}, \pi_\theta\big)$; Safety threshold ϵ.

$p_a \leftarrow \textsc{Extract}(a_i)$ ▷ Extract *action* predicates
$\mathcal{C}_{\theta_a}^{p_a} = \big(\mathcal{P}_{p_a}, R_{p_a}, \theta_a\big) \leftarrow \textsc{Retrieve}(p_a, \mathcal{G}_{\text{ASPM}})$
$\mathcal{V}_s = \{p_s^i : v_s^i\} \leftarrow \emptyset$ ▷ Initialize predicate-value map
for each rule $r = [\mathcal{P}_r, T_r, \phi_r, t_r] \in R_{p_a}$ **do**
  $\mathcal{W}_r \leftarrow \textsc{RetrieveWorkflow}(r, p_a)$
  while $\exists p_s \in \mathcal{P}_r$ s.t. $\mathcal{V}_s[p_s]$ is not assigned **do**
    $A_s \leftarrow \textsc{Plan}(\mathcal{W}_r, r, \mathcal{P}_r)$ ▷ Generate an action plan with shielding operations (e.g., Search, Check)
    for each step t_s^i in action plan A_s **do**
      $o_s^i \leftarrow \textsc{Execute}(t_s^i, \mathcal{H}_{<i}, o_i)$ ▷ Get step result
      $\mathcal{V}_s[p_s] \leftarrow \textsc{Parse}\big(o_s^i\big), p_s \in \mathcal{P}_r$ ▷ Attempt to assign a truth value to any unassigned predicates
    end for
  end while
  $l_r \leftarrow \textsc{Verify}(r, \mathcal{V}_s)$ ▷ Run formal verification
end for
$\epsilon_s \leftarrow P_\theta\big(\mu_{p_a=1}\big) - P_\theta\big(\mu_{p_a=0}\big)$ ▷ Calculate safety condition via Eq. (3)
if $\epsilon_s \geq \epsilon$ **then**
  $l_s \leftarrow 1$ ▷ Action p_a is safe
else
  $l_s \leftarrow 0$ ▷ Action p_a is unsafe
end if
return $\big(l_s, V_s, T_s\big)$ ▷ Return safety label, violated rules, textual explanation

Algorithm 2 ASPM Structure Optimization

Require: Predicate set $\mathcal{P} = \{\mathcal{P}_a, \mathcal{P}_s\}$; Rule set $\mathcal{R} = \{\mathcal{R}_a, \mathcal{R}_p\}$; Embedding model $\mathcal{E}$; Clustering algorithm $\mathcal{C}$; Refinement budget N_b; Max iterations M_it; Surrogate LLM; Graph $G = (\mathcal{P}, E)$ with initial edge weights E.

Initialize vagueness score for each predicate $\mathcal{V}_p, p \in \mathcal{P}$
$\mathcal{V}_r = \max\{\mathcal{V}_{p_1}, \ldots, \mathcal{V}_{p_{|\mathcal{P}_r|}}\}, \mathcal{P}_r \subseteq \mathcal{P}$ ▷ Compute vagueness score for each rule
Initialize a max-heap $\mathcal{U} \leftarrow \big\{(\mathcal{V}_r, r) \;\big|\; r \in \mathcal{R}\big\}$
$n \leftarrow 0$ ▷ Count how many refinements have been done
for $m = 1$ to M_it **do**
  changed $\leftarrow$ false ▷ Tracks if any update occurred in this iteration
  while $\mathcal{U} \neq \emptyset \;\wedge\; n \leq N_\text{b}$ **do**
    $(\_, r) \leftarrow \text{HeapPop}(\mathcal{U})$ ▷ Pop the most *vague* rule
    if LLM_verifiable(r) = false **then**
      $r_\text{new} \leftarrow \text{LLM\_refine}\big(r,\ \mathcal{P}_r\big)$ ▷ Refine rule r to be *verifiable*; update its predicates if needed
      Update $\mathcal{R}$: replace r with r_new
      Update $\mathcal{P}$: if r_new introduces or revises predicates
      Recompute $\mathcal{V}_p$ for any changed predicate p in r_new
      Recompute $\mathcal{V}_{r_\text{new}} = \max\{\mathcal{V}_p \mid p \in \mathcal{P}_{r_\text{new}}\}$
      Push $(\mathcal{V}_{r_\text{new}},\ r_\text{new})$ into $\mathcal{U}$
      $n \leftarrow n + 1$
      changed $\leftarrow$ true
    end if
  end while
  $\mathcal{K} \leftarrow \mathcal{C}(G)$ ▷ Cluster predicates in G to prune redundancy
  for each cluster $C \in \mathcal{K}$ **do**
    $p_\text{merged} \leftarrow \text{LLM\_merge}\big(C,\ \mathcal{R}\big)$ ▷ Merge similar predicates/rules in C if beneficial
    if $p_\text{merged} \neq \emptyset$ **then**
      Update G: add p_merged, remove predicates in C
      Update $\mathcal{R}$ to replace references of predicates in C with p_merged
      Recompute $\mathcal{V}_{p_\text{merged}}$ and any affected $\mathcal{V}_r$
      Push updated rules into $\mathcal{U}$ by their new $\mathcal{V}_r$
      changed $\leftarrow$ true
    end if
  end for
  if changed = false **then**
    break ▷ No more refinements or merges
  end if
end for
return ASPM $\mathcal{G}_\text{ASPM}$ with optimized structure and randomized weights

Algorithm 3 ASPM TRAINING PIPELINE

Require: Rule set $\mathcal{R}$; *state predicates* $\mathcal{P}_s$ and *action predicates* $\mathcal{P}_a$; similarity threshold θ; number of clusters k.

$A \in \{0,1\}^{|\mathcal{P}_s| \times |\mathcal{P}_s|} \leftarrow \mathbf{0}$ ▷ Initialize adjacency matrix

$A_{ij} \leftarrow 1$ if (p_s^i, p_s^j) co-occur in any rule OR $\text{cosSim}\big(\text{emb}(p_s^i),\ \text{emb}(p_s^j)\big) \geq \theta$; else 0. ▷ Build adjacency matrix

labels $\leftarrow$ SPECTRALCLUSTERING$(A,\ k)$ ▷ Cluster the state predicates into k groups

for $\ell = 1$ **to** k **do**

    $C_p^\ell \leftarrow \{p_s \mid \text{labels}[p_s] = \ell\}$ ▷ Form predicate clusters $\mathcal{C}_p$

end for

for each pair (p_s^i, p_s^j) **that co-occur do**

    if $\text{labels}[p_s^i] \neq \text{labels}[p_s^j]$ **then**

        $\mathcal{C}_p^\ell \leftarrow \mathcal{C}_p^\ell \cup \mathcal{C}_p^m$ s.t. $p_s^i \in \mathcal{C}_p^\ell, p_s^j \in \mathcal{C}_p^m$ ▷ If two co-occurring predicates appear in different clusters, merge them

    end if

end for

for $\ell = 1$ **to** k' **do**

    $C_r^\ell \leftarrow \{r_s \mid p_s \in C_p^\ell\}$ ▷ Group rules which share state predicates in the same cluster

end for

$\mathcal{G}_{\text{ASPM}} \leftarrow \varnothing$ ▷ Initialize ASPM as an empty dictionary with actions as keys

for each $p_a \in \mathcal{P}_a$ **do**

    for each rule cluster $C_r^\ell \in \mathcal{C}_r$ **do**

        for each rule $r \in C_r^\ell$ **do**

            if $p_a^r \in r$ **then**

                $\mathcal{G}_{\text{ASPM}}[p_a] = \mathcal{G}_{\text{ASPM}}[p_a] \cup C_r^\ell$ ▷ Associate action circuits with any relevant rule clusters

                break

            end if

        end for

    end for

end for

for each action circuit $\mathcal{C}_{\theta_a}^{p_a}$ **do**

    for each rule $r \in \mathcal{C}_{\theta_a}^{p_a}$ **do**

        Initialize rule weight θ_r randomly

    end for

    for epoch = 1 **to** max epochs **do**

        for $i = 1$ **to** N **do**

            Compute $P_\theta(\mu_{p_a=1}^{(i)})$ and $P_\theta(\mu_{p_a=0}^{(i)})$ ▷ Run probabilistic inference to obtain corresponding safety probabilities via Eq. (1)

            Compute loss $\mathcal{L}(\theta)$ ▷ Calculate loss w.r.t. the groundtruth labels via Eq. (2)

            Update θ using gradient descent

        end for

    end for

end for

return Action-based safety policy model $\mathcal{G}_{\text{ASPM}}$ with trained weights