<u>Definitions and rules:</u>

a – a lifted action.

O- an ordered set of objects
$< a, O >$ - a **grounding** of a if O satisfies $param(a)$ the parameters of a.

L_a - represent the parameter bound literals of an action a.

s, s'- pre state and post state respectively.

$bindP^{-1}\ (F,\ O)$ - $were\ F\ is\ a\ set\ of\ parameter - bound\ literals.$
$the\ function\ returns\ groundings\ of\ F\ in\ respect\ to\ the\ order\ of\ O.$

$bindP\ (f,\ O)$ - $were\ f\ is\ a\ set\ of\ parameter - grounded\ literals.$
$the\ function\ returns\ all\ parameter\ bound\ literals\ in\ respect\ to\ the\ order\ of\ O.$

$pre^*(a), eff^{*-}(a), eff^{*+}(a)\ as\ the\ real\ action\ model\ precodinition\ and\ effect\ of\ the\ action\ a$

<u>Init:</u>
$pre(a) = L_a$ # a set of all predicates who haven't been ruled out as precondition

$pre_?(a) = \emptyset$ # a set of sets containing subsets that each has 1 predicate that must be a precondition

$eff^+(a) = \emptyset$ # all observed predicates who are surely an add effect

$eff^-(a) = \emptyset$ # all observed predicates who are surely a remove effect

$eff^{?+}(a) = L_a$ # all predicates who can possibly be an add effect but yet to be determined as effect

$eff^{?-}(a) = L_a$ # all predicates who can possibly be a remove effect but yet to be determined as effect

<u>Update if a is applicable is state s and $app(a, s) = s'$:</u>

$pre(a) = pre(a) \cap bindP^{-1}(s, O)$

$eff^+(a) = eff^+(a) \cup bindP(s' \setminus s, O)$ # Surely add effect

$eff^-(a) = eff^-(a) \cup bindP(s \setminus s', O)$ # Surely delete effect

$eff^{?+}(a) = eff^{?+}(a) \cap bindP(s \cap s', O)$
Keep all parameter bound literals that we couldn't observe as add effects because they were true in pre and post states. If they are add-effect then they are in $eff^+(a)$, if they are surely not add-effect then they were neg on post state.

$eff^{?-}(a) = eff^?(a) \setminus bindP(s \cup s', O)$ # Remove all predicates that were positive in pre or post states, if it was positive in pre state, than either it was removed to delete effect and then it is observed and found in $eff^-(a)$ or it was true in post state and then it is not a delete effect.

$pre_?(a) = \{B \cap bindP(s, O)\ |B \in pre_?(a)\}$

<u>Update if a is not applicable is state s then $pre(A) \nsubseteq s$:</u>

$pre_?(a) = pre_?(a) \cup \{\ \{pre(a) \setminus bindP(s, O)\}\ \}$ # a set of sets containing subsets that each has 1 predicate that must be a precondition.

Action choosing methodology:

Grounded Action failure\applicability probability in state s:

$define$: $app(a, O, s)$=1 if a is applicable with arguments O from state s, $else$ 0.
we know that:

1) $\forall B \in pre_?(a),\ |B \cap pre^*(a)| \geq 1$
2) $B \cap bindP(s, O) = \emptyset \rightarrow app(a, O, s) = 0$

$$cnf_{pre?(a)} = \bigwedge_{B \in pre_?(a)} \left(\bigvee_{l \in B} x_l \right)$$

$$cnf_{pre?(a),O,s} = cnf_{pre?(a)} \bigwedge_{l \in (\bigcup pre_?(a)) \setminus bindP(s,O)} (\neg x_l)$$

$notice\ that\ for\ (a, O)\ to\ be\ applicable\ in\ s, cnf_{pre?(a),O,s}\ must\ be\ satisfied.$
$due\ to\ the\ fact\ that\ the\ cnf_{pre?(a),O,s}\ is\ over\ the\ same\ set\ of\ variables\ as\ cnf_{pre?(a)}\ and\ that$
$more\ restrctions\ are\ introduced, we\ know\ that: 0 \leq \dfrac{|SAT(cnf_{pre?(a),O,s})|}{|SAT(cnf_{pre?(a)})|} \leq 1$

$SAT(cnf_{pre?(a)}) = \{sol|\ sol\ is\ a\ satisfying\ assignment\ for\ cnf_{pre?(a)}\}$

$SAT(cnf_{pre?(a),O,s}) = \{sol|\ sol\ is\ a\ satisfying\ assignment\ for\ cnf_{pre?(a)}\}$

We know that our sample space- Ω is all possible assignment for $\bigwedge_{l \in L_a}(x_l \vee \neg x_l)$.

And since $SAT(cnf_{pre?(a),O,s}) \subseteq SAT(cnf_{pre?(a)}) \subseteq \mathrm{SAT}(\bigwedge_{l \in L_a}(x_l \vee \neg x_l))$

$pr(app(a, O, s) = 1) = \dfrac{|SAT(cnf_{pre?(a),O,s})|}{|SAT(cnf_{pre?(a)})|}$

$and\ for\ pre_?(a) = \emptyset\ we\ can\ say\ that\ given\ probabillity\ space\ (\Omega, \mathcal{F}, P)$

$$pr(app(a, O, s) = 1) = \begin{cases} 1, & pre_?(a) = \emptyset \\ \dfrac{|SAT(cnf_{pre?(a),O,s})|}{|SAT(cnf_{pre?(a)})|}, & pre_?(a) \neq \emptyset \end{cases}$$

Alternatively, instead of using $pre_?(a)$ as sets. I can initialize it to be at first

$$cnf_{pre?(a)} = \bigwedge_{l \in L_a} (x_l \vee \neg x_l)$$

and for each failed application of $(a, O)\ in\ s$,

$cnf_{pre?(a)} = cnf_{pre?(a)} \wedge (\bigvee_{l \in pre(a) \setminus bindP(s,O)} x_l)$

preconditions gained knowledge:

preconditions gained knowledge if successfully applying $(a, O) from\ s$:

Online Learning of preconditions is tricky, we sometimes can't know if the action is applicable when $pre(a) \nsubseteq bindP^{-1}(s, O)$.

$if\ \ a\ is\ applicable\ in\ s\ then$:

we reduce $|pre(a)|\ \ \ to\ \ |pre(a) \setminus bindP(s, O)|\ parameter\ bound\ literrals.$
means we gain $|pre(a) \setminus bindP(s, O)|$ parameter bound literals who are surely not a precondition. Note the gained information as:
$preAppPotential_{(a,o,s)} = |pre(a) \setminus bindP(s, O)|$

gained knowledge in case of fail execution of action (a, O) from state s:

we want to countify the knowledge gained by failing to execute an action.
mark $preFailPotential_{(a,o,s)}$ the knowledge gained by failing to execute (a, o) from s.
we know that for every $pre^*(a) \subseteq pre(a)$. By failing to execute (a, o) from state s.
we update $pre_?(a) = pre_?(a) \cup \{\ \{pre(a) \setminus bindP^{-1}(s, O)\}\ \}$.
meaning we reduced our set of solution of the constructed $cnf_{pre?(a)}$.
well represent the gained data as the relative amount of solutions we reduced from $cnf_{pre?(a)}$:
given $pre'_?(a) = pre_?(a) \cup \{\ \{pre(a) \setminus bindP^{-1}(s, O)\}\ \}$ is the expected $pre_?(a)$ set if failing,
mark $cnf'_{pre?(a)}\ as\ the\ constructed\ cnf\ formula\ for\ pre'_?(a)$.
so $preFailPotential_{(a,o,s)} = 1 - \frac{|SAT(cnf_{pre?(a)})| - |SAT(cnf'_{pre?(a)})|}{2^{|L_a|}}$

1) We can see that:
   $\exists\{l\} = B \in pre_?(a)\ \ s.t\ |B| = 1 \rightarrow\ l \in pre^*(a)$
   and with enough negative sample on some domains we can achieve a learning state where:
   $\forall l \in pre(a)\ , \{l\} \in pre_?(a)$. meaning we are certain that we learned the preconditions fully and are certain that each literal of the learned preconditions, is indeed a precondition in the true action model.

Effects gained knowledge: TODO: address injective binding problem \ using equality predicate

The sum of $|eff^{?+}(a)|$, $|eff^{?-}(a)|$ is the number of predicates we are not certain are an effect. And therefore, we want to minimize them. We reach the actual model effects of action a when $|eff^{?+}(a)|$, $|eff^{?-}(a)| = 0$. Meaning there is a measure for how much we learned about effects. $grade(eff(a)) = \frac{|L_a| - |eff^{?+}(a) \cup eff^{?-}(a)|}{|L_a|}$. Notice that when $grade(eff) = 1$ we classified all literals as add\delete effect or ruled them out as an effect.

1) If an action a applied successfully from state s, we can say that:
   a. if $l \notin s, l \in eff^{?+}(a)$, then it can be ruled out as an add effect or be observed as one.
   $eff^{+}Potential(s)_{(a,O,s)} = \frac{|eff^{?+} \setminus bindP(s,O)|}{|L_a|}$
   b. if $l \in s,\ l \in eff^{?-}(a)$, then it can be ruled out as a delete effect or be observed as one. $eff^{-}Potential_{(a,O,s)} = \frac{|eff^{?-} \cap bindP(s,O)|}{|L_a|}$.

   From a, b we can know for certain how much we will get closer to classifying all predicates for the effects as surely add, delete or not an effect.

   Let's grade the total knowledge gained for effects as the difference of effect grade before taking the action and after it, out of all parameter bound literals of a.
   $effPotential_{(a,O,s)} = eff^{-}Potential_{(a,O)} + eff^{+}Potential_{(a,O)} = \frac{|eff^{?-} \cap bindP(s,O)| + |eff^{?+} \setminus bindP(s,O)|}{|L_a|}$

Reward:

$sucPotential$ is the gained-knowledge from applying (a, O) from s.

$sucPotential_{(a,O,s)} = effPotential_{(a,O,s)} + preAppPotential_{(a,o,s)}$

$$f((a,O,s)) = \begin{cases} sucPotential, & app(a,O,s) = 1 \\ preFailPotential_{(a,o,s)}, & app(a,O,s) = 0 \end{cases}$$

$$E[X_{(a,O,s)}] = pr(app(a,O,s) = 1) \cdot sucPotential_{(a,O,s)} + pr(app(a,O,s) = 0) \cdot preFailPotential_{(a,o,s)}$$

Action choosing:

option1: our agent will make a greedy choice of action (a, O) to execute s.t:

$$\max_{(a,O) \in grounded\ actions} \{ (a,O) \mid E[X_{(a,O,s)}] \}$$

Option 2:

Define $W: GroundedActions \times S \to R$
$w(a,O,s) = E[X_{(a,O,s)}]$
$\Omega = Grounded - actions, \quad \mathcal{F} = Grounded - actions,$
$P_s(choose(a,O)) = \frac{w(a,O,s)}{\sum_{\forall a,O} w(a,O,s)}$

TODO list for thesis:

- algorithm:
    - organize initialization and update methods in algorithm form
    - add probability calculations
    - Add knowledge gain calculation at each iteration.
    - add mean calculation at each iteration.
    - add action model export at each iteration.
- Proof of consistency for both complete and sound models exported at each iteration.
- Try and find ratio of learning in the worst case of action choosing to bound the difference between the optimal choices and the agents choices.
- Build python project based on sam Framework and add active learning options:
    - How to enable active learning?
        - uml of class diagram
        - flowchart of class diagram
        - use Val and Fast-Downwards
        - Use simple domains first
- Build experiment:
    - Choose domain to experiment on
    - Expected outcome is less actions taken by agent for learning a domain.
    - I can implement Olam and extend the action choosing to my method. And comparing both.