

Expressing and Exploiting the Common Subgoal Structure of Classical Planning Domains Using Sketches

Anonymous

Appendix

In this section, we provide proofs for the theorems that claim the sketches are well-formed and have bounded width, and describe the computation of the features.

Proofs for Sketches

Floortile

Theorem 1. *The sketch for the Floortile domain is well-formed and has sketch width 2.*

Proof. Recall that a sketch is well-formed if it uses goal-separating features and is terminating. The features Φ are goal separating because the feature valuation $g = 0$ holds in state s iff s is a goal state. The sketch R_Φ is terminating because r decreases the numerical feature g and no other rule increases g .

It remains to show that the sketch width is 2. Consider a Floortile instance P with states S . Consider states $S_1 \subseteq S$ where rule r is applicable, i.e., states where some tile t must be painted in some color c . Furthermore, if $(f(s), f(s'))$ is compatible with r then s' is either a goal state or r is applicable in s' and thus we have $s' \in S_1$. With $G_r(s)$ we denote the subgoal states of r in some state s , i.e., states where t is painted in color c . It remains to show that $G_r(s)$ is implied with width at most 2. We do a three-way case distinction over all states S_1 where r is applicable.

First, consider states $S_1^1 \subseteq S_1$ where some robot a on tile t_1 that is configured to color c , can move to tile t_n above or below t to paint it. The singleton tuple $\text{painted}(t, c)$ implies $G_r(s)$ in $s \in S_1^1$ in the admissible chain that consists of moving a from t_1 to t_n , while decreasing the distance to t_n in each step, and painting t , i.e.,

$$(\text{robot-at}(a, t_1), \dots, \text{robot-at}(a, t_n), \text{painted}(t, c)).$$

Second, consider states $S_1^2 \subseteq S_1$ where the robot a must reconfigure its color from c' to c before painting. The tuple $(\text{robot-at}(a, t_n), \text{painted}(t, c))$ implies $G_r(s)$ in $s \in S_1^2$ in the admissible chain that consists of reconfiguring the color,

and then moving closer and painting as before, i.e.,

$$\begin{aligned} &((\text{robot-at}(a, t_1), \text{robot-has}(a, c')), \\ &(\text{robot-at}(a, t_1), \text{robot-has}(a, c)), \dots, \\ &(\text{robot-at}(a, t_n), \text{robot-has}(a, c)), \\ &(\text{robot-at}(a, t_n), \text{painted}(t, c))). \end{aligned}$$

We observe that reconfiguring requires an admissible chain of size 2 because of serializing the reconfiguring and the moving part. Therefore, in the following case, we assume that the robot must reconfigure its color.

Third, consider states $S_1^3 \subseteq S_1$ where robot a is standing on t and there is a sequence of robots a_1, \dots, a_n such that a can only paint t if each a_1, \dots, a_n moves in such a way that tile t' above or below t becomes clear. Using the fact that a rectangular portion inside a rectangular grid has to be painted, it follows that the set of tiles that must not be painted is pairwise connected. Therefore, we can move each robot a_i from its current tile t'_i to t_i in such a way that after moving each robot, tile t becomes clear. The tuple $(\text{robot-at}(a, t_n), \text{painted}(t, c))$ implies $G_r(s)$ in $s \in S_1^3$ in the admissible chain that consists of moving each robot a_i from t'_i to t_i in such a way that moving all of them clears tile t' , followed by moving a to t' , and painting t , i.e.,

$$\begin{aligned} &((\text{robot-at}(a, t), \text{robot-has}(a, c')), \\ &(\text{robot-at}(a, t), \text{robot-has}(a, c)), \\ &(\text{robot-at}(a_1, t_1), \text{robot-has}(a, c)), \dots, \\ &(\text{robot-at}(a_n, t_n), \text{robot-has}(a, c)), \\ &(\text{robot-at}(a, t'), \text{robot-has}(a, c)), \\ &(\text{robot-at}(a, t'), \text{painted}(t, c))). \end{aligned}$$

We obtain sketch width 2 because all tuples in admissible chains have size of at most 2. \square

Grid

Theorem 2. *The sketch for the Grid domain is well-formed and has sketch width 1.*

Proof. The features Φ are goal separating because the feature valuation $k = 0$ holds in state s iff s is a goal state. We show that the sketch is terminating by iteratively eliminating rules: r_1 decreases l which no other rule increases, so we eliminate r_1 and mark l . Now r_2 can be eliminated because

it decreases k which no remaining rule increases. We can now eliminate $r_3 = C \mapsto E$ because it changes the Boolean feature o and the only other remaining rule $r_4 = C' \mapsto E'$ may restore the value of o , but this can only happen finitely often, since l is marked and $l > 0 \in C$ and $l = 0 \in C'$. Now only r_4 remains and we can eliminate it since it changes t , which is never changed back.

It remains to show that the sketch width is 1. Consider any Grid instance P with states S . If there are closed locks in the initial state then rule r_1 is applicable to open them. Furthermore, if all locks are open, then rule r_2 is applicable to move keys to their target cell. Note that r_2 decreases the number of unachieved goal atoms until the goal G is satisfied. Therefore it suffices to show that (1) the subgoal $G_{r_1}(s)$ of r_1 in all R -reachable states where only r_1 is applicable is implied with width 1, and (2) the subgoal $G_{r_2}(s)$ of r_2 in all R -reachable states where only r_2 is applicable is implied with width 1. This suffices because termination ensures finite state trajectories so that in all states s where rules $R' \subseteq R$ are applicable with $r_1 \in R'$, it is sufficient to show that $G_r(s)$ is implied with width 1 for some $r \in R'$. Termination ensures in this case that at some point we reach a state s where only r_1 is applicable for which we have to show that $G_{r_1}(s)$ is implied with width 1. The same argument holds for r_2 .

We first consider rule r_3 . Intuitively, we show that picking up a key that can be used to open some closed lock has width 1. Consider states $S_1 \subseteq S$ where r_3 is applicable, i.e., states where there is a closed lock and the robot does not hold a key e that can be used to open a closed lock. With $G_{r_3}(s)$ we denote the subgoal states of r_3 in $s \in S_1$, i.e., states where the robot holds e . The tuple $\text{holding}(e)$ implies $G_{r_3}(s)$ in $s \in S_1$ in the admissible chain that consists of changing the position of the robot from the current position c_1 to the position c_n of e ordered by the distance to c_n , and followed by exchanging or picking e , i.e., $(\text{at-robot}(c_1), \dots, \text{at-robot}(c_n), \text{holding}(e))$. Note that r_1 is the only applicable rule in $G_{r_3}(s)$.

Next, we consider rule r_1 . Intuitively, we show that opening a closed lock has width 1. Consider states $S_2 \subseteq S$ where r_1 is applicable and the robot holds a key e that can be used to open a closed lock d . We can transform states where the robot holds no key into a state from S_2 by letting it pick a key with width 1 (see above). With $G_{r_1}(s)$ we denote the subgoal states of r_1 in $s \in S_2$, i.e., states where d is open. The tuple $\text{open}(d)$ implies $G_{r_1}(s)$ in $s \in S_2$ in the admissible chain that consists of changing the position of the robot from its current position c_1 to a position c_n next to lock d ordered by the distance to c_n , i.e., $(\text{at-robot}(c_1), \dots, \text{at-robot}(c_n), \text{open}(d))$. It remains to show that if all locks are open then moving keys has width 1.

Next, we consider rule r_4 . Intuitively, we show that picking up a key that is not at its target cell has width 1. Consider states $S_3 \subseteq S$ where r_4 is applicable, i.e., states where all locks are open and the robot does not hold a misplaced key. With $G_{r_4}(s)$ we denote the subgoal states of r_4 in $s \in S_3$, i.e., states where the robot holds e . The tuple $\text{holding}(e)$ implies $G_{r_4}(s)$ in $s \in S_3$ in the admissible chain that con-

sists of changing the position of the robot from the current position c_1 to the position c_n of e ordered by the distance to c_n , and followed by exchanging or picking e , i.e., $(\text{at-robot}(c_1), \dots, \text{at-robot}(c_n), \text{holding}(e))$. Note that r_2 is the only applicable rule in $G_{r_4}(s)$.

Finally, we consider rule r_2 . Intuitively, we show that moving a key to its target cell has width 1. Consider states $S_4 \subseteq S$ where r_2 is applicable and the robot holds a misplaced key e . As before, we can transform states $s' \notin S_4$ into such a state s by picking up e with width 1. With $G_{r_2}(s)$ we denote the subgoal states of r_2 in $s \in S_4$, i.e., states where e is at its target cell. The tuple $\text{at}(e, c_n)$ implies $G_{r_2}(s)$ in $s \in S_4$ in the admissible chain that consists of changing the position of the robot from its current position c_1 to the key's target cell c_n ordered by the distance to c_n , followed by exchanging or dropping the key at c_n , i.e., $(\text{at-robot}(c_1), \dots, \text{at-robot}(c_n), \text{at}(e, c_n))$.

We obtain sketch width 1 because all tuples in admissible chains have size of at most 1. \square

Barman

Theorem 3. *The sketch for the Barman domain is well-formed and has sketch width 2.*

Proof. The features Φ are goal separating because $g = 0$ holds in state s iff s is a goal state. We show that the sketch is terminating by iteratively eliminating rules: first, we eliminate r_4 because it decreases the numerical feature g that no rule increases. Next, rules r_1 and r_2 can be eliminated because both change a Boolean feature that no remaining rule changes in the opposite direction. Last, we eliminate the remaining rule r_3 because it decrements the numerical feature u .

It remains to show that the sketch width is 2. Consider any Barman instance P with states S . Note that for any pair of feature valuations $(f(s), f(s'))$ compatible with r_4 , $s \in S$ is a non-goal state and $s' \in S$ is either a goal or a non-goal state. Thus, r_4 is applicable in any non-goal initial state, and it suffices to show that the subgoal $G_{r_4}(s)$ in every R -reachable non-goal state s where only r_4 is applicable is implied with width at most 2. This suffices because termination ensures finite state trajectories such that in all states s where there are rules $R' \subseteq R$ applicable with $r_4 \in R'$, it is sufficient to show that $G_r(s)$ is implied with width 2 for some $r \in R'$. Termination ensures in this case that at some point we reach a state s where only r_4 is applicable for which we have to show that $G_{r_4}(s)$ is implied with width 2.

We first consider rule r_3 . Intuitively, we show that shots are cleaned with width at most 1. Consider all states $S_1 \subseteq S$ where r_3 is applicable, i.e., states where there is a used shot g such that $\text{used}(g, b)$ holds for some beverage b that is not supposed to be in g according to the goal description. With $G_{r_3}(s)$ we denote the subgoal states of r_3 in $s \in S_1$, i.e., states where g is clean. We do a case distinction over states S_1 . First, consider states $S_1^1 \subseteq S_1$ where the barman is holding g in hand h . The tuple $\text{clean}(g)$ implies $G_{r_3}(s)$ for all $s \in S_1^1$ in the admissible chain that consists of cleaning g , i.e., $(\text{holding}(h, g), \text{clean}(g))$. Second, consider states $S_1^2 \subseteq S_1$ where the barman must grasp g with empty

hand h first. The same tuple $clean(g)$ implies $G_{r_3}(s)$ for all $s \in S_1^2$ in the admissible chain that consists of picking g , and cleaning g , i.e., $(ontable(g), holding(h, g), clean(g))$. Last, consider states $S_1^3 \subseteq S_1$ where the barman must exchange g' with g in hand h first. The same tuple $clean(g)$ implies $G_{r_3}(s)$ for all $s \in S_1^3$ in the admissible chain that consists of putting down g' , picking up g , cleaning g , i.e., $(holding(h, g'), ontable(g'), holding(h, g), clean(g))$. It also follows that we can reduce the set of R -reachable states in our analysis to those where the container is already grasped if only a single container is affected.

Next, we consider rule r_1 . Intuitively, we show that filling the first ingredient into the shaker for producing a required cocktail has width at most 2. Consider states $S_2 \subseteq S$ where r_1 is applicable and required shots are clean, i.e., states where no ingredient i_1 consistent with the first part of some unserved cocktail c 's recipe is in the shaker t . We do not need to consider states where required shots are not clean because a shot can be cleaned with width 1 (see above). With $G_{r_1}(s)$ we denote the subgoal states of r_1 in s , i.e., states where an ingredient i_1 consistent with the first recipe part of some unserved cocktail c is inside t . The tuple $(contains(t, i_1), shaker-level(t, l1))$ implies $G_{r_1}(s)$ in the admissible chain that consists of cleaning t , putting down t , picking a clean shot g , filling i_1 into g using the corresponding dispenser, and pouring g into t , i.e.,

$((holding(h, t), shaker-level(t, l2)),$
 $(holding(h, t), empty(t)), (holding(h, t), clean(t)),$
 $(ontable(t), clean(t)), (holding(h, g), clean(t)),$
 $(contains(g, i_1), used(g, i_1)),$
 $(contains(t, i_1), shaker-level(t, l1))).$

Note that rule r_2 is applicable in $G_{r_1}(s)$. This ensures that rule r_2 defines the next subgoal for producing the cocktail and bounds the width for filling the second ingredient into the shaker.

Next, we consider rule r_2 . Intuitively, we show that filling the second ingredient into the shaker for producing a required cocktail has width at most 1. Consider states $S_3 \subseteq S$ where r_2 is applicable and required shots are clean, i.e., states where the first ingredient consistent with the recipe of an unserved cocktail c is in the shaker t , and required shots are clean because a shot can be cleaned with width 1 (see above). With $G_{r_2}(s)$ we denote the subgoal states of r_2 in s , i.e., states where an ingredient i_2 is inside t such that both ingredients in t are consistent with the recipe of an unserved cocktail c . The tuple $(contains(t, i_2), shaker-level(t, l2))$ implies $G_{r_2}(s)$ in the admissible chain that consists of putting down t , grasping g , filling i_2 into g using the corresponding dispenser, and pouring g into t , i.e.,

$((holding(h, t), shaker-level(t, l1)),$
 $(ontable(t), shaker-level(t, l1)),$
 $(holding(h, g), clean(g)),$
 $(contains(g, i_2), used(g, i_2)),$
 $(contains(t, i_2), shaker-level(t, l2))).$

Finally, we consider rule r_4 , where we show intuitively that serving a beverage has width at most 1. We do a

case distinction over all states S_4 where r_4 is applicable, i.e., states where there is an unserved ingredient or an unserved cocktail. First, consider states $S_4^1 \subseteq S_4$ where there is an unserved ingredient i . $G_{r_4}^1(s)$ is the set of subgoal states for r_4 in $s \in S_4^1$ where i is served. The tuple $contains(g, i)$ implies $G_{r_4}^1(s)$ in the admissible chain that consists of filling i into g using the corresponding dispenser, i.e., $(clean(g), contains(g, i))$. Last, consider states $S_4^2 \subseteq S_4$ where there is an unserved cocktail c , respective ingredients are in the shaker using the results of rule r_1, r_2 , and required shots are clean using the results of rule r_3 . With $G_{r_4}^2(s)$ we denote the subgoal states of r_4 in $s \in S_4^2$ where c is served. The tuple $contains(g, c)$ implies $G_{r_4}^2(s)$ in the admissible chain that consists of putting down g (or any other shot) because shaking requires only the shaker t to be held, shaking t , and pouring t into g , i.e.,

$(holding(h, g), ontable(g), contains(t, c), contains(g, c))$

We obtain sketch width 2 because all tuples in admissible chains have size of at most 2. \square

Childsnack

Theorem 4. *The sketch for the Childsnack domain is well-formed and has sketch width 1.*

Proof. The features are goal separating because the feature valuations $c_g = 0$ and $c_r = 0$ hold in state s iff s is a goal state. We show that the sketch is terminating by iteratively eliminating rules: r_5 decreases the numerical feature c_g which no other rule increments, so we eliminate r_5 and mark c_g . Similarly, r_6 decreases the numerical feature c_r which no other rule increments, so we eliminate r_6 and mark c_r . Then rules r_4 changes s^t and no remaining rules changes s^t in the opposite direction, so we eliminate r_4 . Likewise, we eliminate r_3 because it changes s_g^t , which no remaining rule can change back. Last, we eliminate rules r_1 and r_2 because they change s_g^k resp. s^k , and no remaining rule can change the values in the opposite direction.

It remains to show that the sketch width is 1. Consider any Childsnack instance. Note that if there is an unserved gluten-allergic child in the initial state then rules r_1, r_3, r_5 define subgoals for serving a gluten-allergic child. If there is no unserved gluten-allergic child but there is an unserved non-allergic child then rules r_2, r_4, r_6 define subgoals for serving a non-allergic child. In the following, we first show that serving a gluten-free sandwich to a gluten-allergic child has width 1 and deduce the case of serving a non-allergic child from it.

We first consider rule r_1 . Intuitively, we show that producing a gluten-free sandwich has width 1. Consider states $S_3 \subseteq S$ where r_1 is applicable, i.e., states where there is an unserved gluten-allergic child c and there is no gluten-free sandwich available in *kitchen* nor on a tray. With $G_{r_1}(s)$ we denote the subgoal states of r_1 in $s \in S_3$, i.e., states where gluten-free sandwich s is available in *kitchen*. The tuple $no-gluten-sandwich(s)$ implies $G_{r_1}(s)$ in $s \in S_3$ in the admissible chain that consists of producing s , i.e., $(notexists(s)), no-gluten-sandwich(s))$. Note that only r_3 is applicable in subgoal states $G_{r_1}(s)$.

Next, we consider rule r_3 . Intuitively, we show that moving a gluten-free sandwich from the kitchen onto a tray has width 1. Consider states $S_2 \subseteq S$ where r_3 is applicable, i.e., states where there is an unserved gluten-allergic child c and there is a gluten-free sandwich s available in *kitchen*. With $G_{r_3}(s)$ we denote the subgoal states of r_3 in $s \in S_2$, i.e., states where s is on p . The tuple $ontray(s, p)$ implies $G_{r_3}(s)$ in $s \in S_2$ in the admissible chain that consists of moving p from t to *kitchen*, putting s onto p , i.e., $(at(p, t), at(p, kitchen), ontray(s, p))$. Note that only r_5 is applicable in subgoal states $G_{r_3}(s)$.

Next, we consider rule r_5 . Intuitively, we show that serving a gluten-allergic child if there is a gluten-free sandwich is available on a tray has width 1. Consider states $S_1 \subseteq S$ where r_5 is applicable, i.e., states where there is an unserved gluten-allergic child c and there is a gluten-free sandwich s available on a tray p . With $G_{r_5}(s)$ we denote the subgoal states of r_5 in $s \in S_1$, i.e., states where c is served. The tuple $served(c)$ implies $G_{r_5}(s)$ in $s \in S_1$ in the admissible chain that consists of moving p from *kitchen* to t , serving c with s , i.e., $(ontray(s, p), at(p, t), served(c))$. Note that if all gluten allergic children are served in subgoal states $G_{r_5}(s)$ then either G was reached or either r_2 , r_4 or r_6 become applicable if there is an unserved regular child. Otherwise, if there remains an unserved gluten allergic child, then r_1 , r_3 or r_5 become applicable depending on whether there are gluten free sandwiches available in the kitchen or on a tray. As seen before, the subgoal states of r_1 , r_3 and r_5 are implied with width 1.

In the case where all gluten-allergic children have been served (or there is no gluten-allergic child in the first place) and there are only unserved non-allergic children, the problem is very similar to the one we considered above and rules r_2, r_4, r_6 define the corresponding subgoals to serve a non-allergic child. We omit the details but provide the admissible chains that are necessary to conclude the proof: the tuple $served(c)$ implies $G_{r_6}(s)$ in the admissible chain $(ontray(s, p), at(p, t), served(c))$. The tuple $ontray(s, p)$ implies $G_{r_4}(s)$ in the admissible chain $(at(p, t), at(p, kitchen), ontray(s, p))$. The tuple $at-kitchen-sandwich(s)$ implies $G_{r_2}(s)$ in the admissible chain $(notexists(s), at-kitchen-sandwich(s))$.

As a result, we get sketch width 1 because all tuples in admissible chains have size of at most 1. \square

Driverlog

Theorem 5. *The sketch for the Driverlog domain is well-formed and has sketch width 1.*

Proof. The features are goal separating because the feature valuations $p=0, t=0, d_g=0$ hold in state s iff s is a goal state. We show that the sketch is terminating by iteratively eliminating rules: r_3 decreases the numerical feature p that no other remaining rule increments, so we eliminate r_3 and mark p . We can now eliminate $r_5 = C \mapsto E$ because it decreases the numerical feature t and the only other remaining rule $r_2 = C' \mapsto E'$ arbitrarily changes t , but this can only happen finitely many times, since p is marked and

$p=0 \in C$ and $p>0 \in C'$. Next, we can eliminate r_2 because it sets the Boolean feature l and no other remaining rule changes l in the opposite direction. We can now eliminate $r_4 = C \mapsto E$ because it decreases the numerical feature d_t and the only other remaining rule $r_1 = C' \mapsto E'$ arbitrarily changes d_t , but this can only happen finitely many times, since p is marked and $p=0 \in C$ and $p>0 \in C'$. Next, we can now eliminate $r_6 = C \mapsto E$ because it decreases the numerical feature d_g and the only other remaining rule $r_1 = C' \mapsto E'$ arbitrarily changes d_g , but this can only happen finitely many times, since p is marked and $p=0 \in C$ and $p>0 \in C'$. Last, we eliminate the remaining rule r_1 because it sets the Boolean feature b to true.

It remains to show that the sketch width is 1. Consider any Driverlog instance. If there are misplaced packages in the initial state, then rule r_3 decrements the number of misplaced packages. Therefore, we show that moving packages to their target location has width 1. Consider states $S_1 \subseteq S$ where there is a misplaced package p at location c_m with target location c_o . We do a three-way case distinction over all states S_1 and show that moving a package to its target location has width 1. First, consider rule r_1 . Intuitively, we show that boarding some driver into a truck has width 1. Consider states $S_1^1 \subseteq S_1$ where rule r_1 is applicable, i.e., states where there is no driver boarded into any truck. With $G_{r_1}(s)$ we denote the subgoal states of r_1 in $s \in S_1^1$, i.e., states where a driver d is boarded into a truck t . The tuple $driving(d, t)$ implies $G_{r_1}(s)$ in $s \in S_1^1$ in admissible chain that consists of moving d from c_1 to c_n , each step decreasing the distance to c_n , boarding d into t , i.e.,

$$(at(d, c_1), \dots, at(d, c_n), driving(d, t)).$$

Second, consider rule r_2 . Intuitively, we show that loaded a misplaced package into a truck has width 1. Consider states $S_1^2 \subseteq S_1$ where rule r_2 is applicable and where d is boarded into truck t at location l_n , i.e., no misplaced package is loaded, and d is boarded into t at location l_n because boarding has d into t if there is a misplaced package has width 1 (see above). With $G_{r_2}(s)$ we denote the subgoal states of r_2 in $s \in S_1^2$, i.e., states where p is loaded into t . The tuple $in(p, t)$ implies $G_{r_2}(s)$ in $s \in S_1^2$ in the admissible chain that consists of driving t from c_n to c_m , each step decreasing the distance to c_m , loading p into t , i.e.,

$$(at(t, c_n), \dots, at(t, c_m), in(p, t)).$$

Third, consider rule r_3 . Intuitively, we show that moving a package to its target location has width 1. Consider states $S_1^3 \subseteq S_1$ where rule r_3 is applicable and where p and d is in t at c_m , i.e., states where p and d is in t at c_m because loading driver and misplaced package has width 1 (see above). With $G_{r_3}(s)$ we denote the subgoal states of r_3 in $s \in S_1^3$, i.e., states where p is at location c_o . The tuple $at(p, c_o)$ implies $G_{r_3}(s)$ in the admissible chain that consists of driving t from c_m to c_o , each step decreasing the distance to c_o , and unloading p , i.e.,

$$(at(t, c_m), \dots, at(t, c_o), at(p, c_o)).$$

Now, consider states S_2 where all packages are at their respective target location and there is a misplaced truck t at

location l_n with target location l_m . This can either be the case in the initial state or after moving the packages because it requires to use trucks. We do a two-way case distinction over all states S_2 and show that moving a truck to its target location has width 1. Consider rule r_4 . Intuitively, we show that boarding a driver into a misplaced truck without using any truck has width 1. Consider states $S_2^1 \subseteq S_2$ where rule r_4 is applicable, i.e., where there is a driver d at location c_1 with nonzero distance until being boarded into t . With $G_{r_4}(s)$ we denote the subgoal states of r_4 in $s \in S_2^1$, i.e., states where d is one step closer to being boarded into t . There are three possible admissible chains that must be considered. (1) unboarding d from some truck t' , i.e., tuple $at(d, c_1)$ implies $G_{r_4}(s)$ in $s \in S_2^1$ in the admissible chain $(driving(d, t'), at(d, c_1))$, (2) moving d closer to c_n over c_{i-1} to c_i , i.e., tuple $at(d, c_i)$ implies $G_{r_4}(s)$ in $s \in S_2^1$ in the admissible chain $(at(d, c_{i-1}), at(d, c_i))$, and (3) boarding d into t at c_n , i.e., $driving(d, t)$ implies $G_{r_4}(s)$ in $s \in S_2^1$ in the admissible chain $(at(d, c_n), driving(d, t))$. Second, consider rule r_5 . Intuitively, we show that moving a misplaced truck to its target location has width 1. Consider states $S_2^2 \subseteq S_2$ where rule r_5 is applicable and where some driver is boarded into t , i.e., states where d is boarded into t at c_n . With $G_{r_5}(s)$ we denote the subgoal states of r_5 in $s \in S_2^2$, i.e., states where t is at its target location. The tuple $at(t, c_n)$ implies $G_{r_5}(s)$ in $s \in S_2^2$ in the admissible chain that consists of moving t from c_n to c_m , each step decreasing the distance to c_m , i.e., $(at(t, c_n), \dots, at(t, c_m))$.

Now, consider states S_3 where all packages and trucks are at their respective target location and there is a misplaced driver d boarded or unboarded at location l_1 with target location l_n . This can either be the case in the initial state or after moving the packages and trucks. Consider rule r_6 . Intuitively, we show that moving a driver to its target location without using any truck has width 1. With $G_{r_6}(s)$ we denote the subgoal states of r_6 in $s \in S_3$, i.e., states where d is at its target location. There are two possible admissible chains that must be considered. (1) unboarding d at location c_1 , i.e., tuple $at(d, c_1)$ implies $G_{r_6}(s)$ in $s \in S_3$ in the admissible chain $(driving(d, t), at(d, c_1))$, and (2) moving d closer from location c_{i-1} to c_i , i.e., tuple $at(d, c_i)$ implies $G_{r_6}(s)$ in $s \in S_3$ in the admissible chain $(at(d, c_{i-1}), at(d, c_i))$.

We obtain sketch width 1 because all tuples in admissible chains have size of at most 1. Note, when dropping rules r_1 and r_2 , as well as features l and b , the sketch width becomes 2 because we must merge the three admissible chains of the first subproblem. When merging, tuples of size two must be considered, each consisting of a location and whether the driver is driving the truck or whether the package is loaded. \square

Schedule

Theorem 6. *The sketch for the Schedule domain is well-formed and has sketch width 2.*

Proof. The features are goal separating because the feature valuations $p_1 = 0, p_2 = 0, p_3 = 0$ hold in state s iff s is a goal state. We show that the sketch is terminating by iteratively eliminating rules: r_1 decreases the numerical feature

p_1 that no other remaining rule increments, so we eliminate r_1 and mark p_1 . r_2 decreases the numerical feature p_2 that no other remaining rule increments, so we eliminate r_2 and mark p_2 . r_3 decreases the numerical feature p_3 that no other remaining rule increments, so we eliminate r_3 and mark p_3 . Last, we eliminate the remaining rule r_4 because it sets the Boolean feature o to false.

It remains to show that the sketch width is 2. Consider any Schedule instance. First, consider states $S_1 \subseteq S$ where r_4 is applicable, i.e., there is either a scheduled object or a machine occupied. This can be the case in the initial state or if an object is scheduled to be processed by a machine. With $G_{r_4}(s)$ we denote the subgoal states of r_4 in $s \in S_1$, i.e., states where no object is scheduled and no machine is occupied while not removing any other achieved goal atom. The tuple $\neg scheduled(a)$ implies $G_{r_4}(s)$ in $s \in S_1$ in the admissible chain that consists of the action that performs a single time step, i.e., $(scheduled(a), \neg scheduled(a))$.

Now, consider states $S_2 \subseteq S$ where r_1 is applicable and there is no object scheduled and no occupied machine, i.e., states where there is an object a that has shape x that is not the shape y mentioned in the goal, and there is no object scheduled and no occupied machine because it can be achieved with width 1 (see above). With $G_{r_1}(s)$ we denote the set of subgoal states of r_1 in $s \in S_2$, i.e., states where a has shape y while not removing any other achieved goal atom. The tuple $(shape(a, z), temperature(x, t))$ implies $G_{r_1}(s)$ in $s \in S_2$ in the admissible chain that consists of changing the shape with procedures that do not affect the temperature of a , i.e.,

$$((shape(a, y), temperature(x, t)), \\ (shape(a, z), temperature(x, t)))$$

Now, consider states $S_3 \subseteq S$ where r_2 is applicable, there is no object scheduled and no machine occupied, and objects have their correct shape, i.e., states where there is an object a that has surface x that is not the surface y mentioned in the goal, there is no object scheduled and no occupied machine because it can be achieved with width 1 (see above), and all objects have their correct shape because changing the shape has width 2 (see above). With $G_{r_2}(s)$ we denote the set of subgoal states of r_2 in $s \in S_3$, i.e., states where a has surface y while not removing any other achieved goal atom. The tuple $(surface-condition(a, z), temperature(x, t))$ implies $G_{r_2}(s)$ in $s \in S_3$ in the admissible chain that consists of changing the surface with procedures that do not affect the temperature of a , i.e.,

$$((surface-condition(a, y), temperature(x, t)), \\ (surface-condition(a, z), temperature(x, t)))$$

Now, consider states $S_4 \subseteq S$ where r_3 is applicable, there is no object scheduled and no machine occupied, objects have their correct shape, and objects have their correct surface, i.e., states where there is an object a that has color x that is not the color y mentioned in the goal, there is no object scheduled and no occupied machine because it can be achieved with width 1 (see above), all objects have their correct shape because changing the shape has width 2 (see

above), and all objects have their correct surface because changing the surface has width 2 (see above). With $G_{r_3}(s)$ we denote the set of subgoal states of r_3 in $s \in S_3$, i.e., states where a has color y while not removing any other achieved goal atom. The tuple $(\text{painted}(a, y), \text{temperature}(x, t))$ implies $G_{r_2}(s)$ in $s \in S_2$ in the admissible chain that consists of changing the surface with procedures that do not affect the temperature of a , i.e.,

$$\begin{aligned} &((\text{painted}(a, y), \text{temperature}(x, t)), \\ &(\text{painted}(a, z), \text{temperature}(x, t))) \end{aligned}$$

Note that r_3 achieved the goal when the color of the last object changes to the color mentioned in the goal.

We obtain sketch width 2 because all tuples in admissible chains have size of at most 2. \square

Feature Grammar

The feature grammar is made up of standard description logics constructors. Description logics (Baader et al. 2003) considers two type of object classes: concepts and roles. Concepts can be interpreted as grounded atoms made up from unary predicates or as objects and roles as grounded atoms made up from binary predicates. These predicates typically come from the first order planning domain. However, classical planning domain can contain predicates with arity higher than two. The following definition of syntax and semantics is based on those given by Francès, Bonet, and Geffner (2021). We redefine primitive concepts and primitive roles in a slightly more expressive way to avoid having to reformulate the domain.

Syntax and Semantics

The set of possible concepts and roles for each state s are inductively defined as follows. Consider concepts C, D and roles R, S and the universe Δ containing all objects in the planning instance.

- A *primitive concept* $p[i]$ is a set of unary predicates made up from predicate p and objects occurring at index i in respective grounded atoms in state s .¹
- A *primitive role* $p[i, j]$ is a set of binary predicates made up from predicate p and objects occurring at index i and j in respective grounded atoms in state s .²
- The *universal concept* \top and the *bottom concept* \perp are concepts with denotations $\top = \Delta, \perp = \emptyset$.
- The concept *union* $C \sqcup D$, *intersection* $C \sqcap D$, *negation* $\neg C$ are concepts with denotations $(C \sqcup D) = C \cup D$, $(C \sqcap D) = C \cap D$, $(\neg C) = \Delta \setminus D$.
- The *existential abstraction* $\exists R.C$ and the *universal abstraction* $\forall R.C$ are concepts with denotations $(\exists R.C) = \{a \mid \exists b : (a, b) \in R \wedge b \in C\}$, $\forall R.C = \{a \mid \forall b : (a, b) \in R \rightarrow b \in C\}$.
- If a is a constant in the domain the *nominal* a is a concept that represents $\{a\}$.

¹often makes it possible to consider higher arity predicates.

²often makes it possible to consider higher arity predicates.

- The role *union* $R \sqcup S$, *intersection* $R \sqcap S$, *complement* $\neg R$ are roles with denotations $(R \sqcup S) = R \cup S$, $(R \sqcap S) = R \cap S$, $\neg R = (\Delta \times \Delta) \setminus R$.
- The *role-value map* $R = S, R \subseteq S$ is a concept with denotation $(R = S) = \{a \mid \forall b : (a, b) \in R \leftrightarrow (a, b) \in S\}$, $(R \subseteq S) = \{a \mid \forall b : (a, b) \in R \rightarrow (a, b) \in S\}$.
- The *composition* $R \circ S$ is a role with denotation $R \circ S = \{(a, c) \mid (a, b) \in R \wedge (b, c) \in S\}$.
- The *inverse* R^{-1} is a role with denotation $R^{-1} = \{(b, a) \mid (a, b) \in R\}$.
- The *restriction* $R|_C$ is a role with denotation $R|_C = R \sqcap (\Delta \times C)$.
- The *identity* $\text{id}(C)$ is a role with denotation $\text{id}(C) = \{(a, a) \mid a \in C\}$.
- The *difference* $C \setminus D, R \setminus S$ is a concept and a role respectively with denotation $C \setminus D = C \sqcap \neg D$, $R \setminus S = R \sqcap \neg S$.³
- The *extraction* $R[i]$ with $i \in \{0, 1\}$ is a concept with denotation $R[0] = \exists R.\top$ and $R[1] = \exists R^{-1}.\top$.⁴

Furthermore, for C and R we have corresponding goal versions denoted by C_g and R_g that are evaluated in the goal of the planning instance instead of the state s as described in Francès, Bonet, and Geffner (2021).

From Concepts and Roles to Features

We define Boolean and numerical features with an additional level of composition as follows. Consider concepts C, D and roles R, S, T .

- *Nonempty*(x) is true iff $|x| > 0$.
- *Empty*(x) is true iff $|x| = 0$.
- *Count*(x) counts the number of elements in x .
- *ConceptDist*(C, R, D) represents the smallest $n \in \mathbb{N}_0$ s.t. there are objects $x_0, \dots, x_n, x_0 \in C, x_n \in D$ and $R(x_i, x_{i+1})$ with $i = 0, \dots, n-1$. If no such n exists then the feature evaluates to ∞ .
- *RoleDist*(R, S, T) represents the smallest $n \in \mathbb{N}_0$ s.t. there are objects x_0, \dots, x_n , there exists some $(a, x_0) \in R, (a, x_n) \in T$, and $R(x_i, x_{i+1})$ with $i = 0, \dots, n-1$. If no such n exists then the feature evaluates to ∞ .
- *SumRoleDist*(R, S, T) := $\sum_{r \in R} \text{RoleDist}(\{r\}, S, T)$ where the sum evaluates to ∞ if any term is ∞ .

Floortile

Consider the set of elements $\{x_1, \dots, x_6\}$ with

$$\begin{aligned} x_1 &\equiv (\text{painted}_g[0, 1] \setminus \text{painted}[0, 1])[0] \\ x_2 &\equiv (\text{left}[0] \sqcup \text{left}[1]) \setminus \text{painted}_g[0] \\ x_3 &\equiv \text{up}[0, 1] \sqcup \text{down}[0, 1] \sqcup \text{id}(\text{left}[0] \sqcup \text{left}[1]) \\ x_4 &\equiv ((x_3|_{x_1})^{-1}|_{x_1})^{-1} \\ x_5 &\equiv ((x_3|_{x_2})^{-1}|_{x_2})^{-1} \end{aligned}$$

³for convenience.

⁴for convenience.

where x_1 is a concept that represents the set of all unpainted tiles, x_2 is a concept that represents the set of all normal tiles that must not be painted, x_3 is a role that describes what tiles are above or below some other tile and the identity, x_4 is a role that describes what unpainted tiles are above or below some unpainted tile, and x_5 is a role that describes what normal tile is above or below some unpainted tile. The features $\Phi = \{v, g\}$ are constructed as follows.

$$v \equiv \text{Empty}(x_1 \setminus (x_4^* \circ x_5))$$

$$g \equiv \text{Count}(x_1)$$

TPP

Consider the set of elements $\{x_1, x_2\}$ with

$$x_1 \equiv (\text{stored}_g[0, 1] \setminus \text{stored}[0, 1])[0]$$

$$x_2 \equiv \text{next}[1]$$

where x_1 is a concept that represents the set of goods of which some quantity remains to be stored, and x_2 is a concept that represents the set of non empty levels. The features $\Phi = \{u, w\}$ are constructed as follows.

$$u \equiv \text{Count}(x_1 \setminus \exists \text{loaded}[0, 2].x_2)$$

$$w \equiv \text{SumRoleDist}(\text{stored}[0, 1], \text{next}[1, 0], \text{stored}_g[0, 1])$$

Barman

Single Shaker Consider the set of elements $\{x_1, x_2, x_3\}$ with

$$x_1 \equiv (\text{contains}_g[0, 1] \setminus \text{contains}[0, 1])$$

$$x_2 \equiv \forall \text{cocktail-part1}[0, 1].\exists \text{contains}[1, 0].\text{shaker-level}[0]$$

$$x_3 \equiv \forall \text{cocktail-part2}[0, 1].\exists \text{contains}[1, 0].\text{shaker-level}[0]$$

where x_1 is a role that describes what beverages remain to be served, x_2 is a concept that represents the set of cocktails where the first ingredient mentioned in its recipe is in the shaker, and x_3 is a concept that represents the set of cocktails where the second ingredient mentioned in its recipe is in the shaker. The features $\Phi = \{g, u, c_1, c_2\}$ are constructed as follows.

$$g \equiv \text{Count}(x_1)$$

$$u \equiv \text{Count}(\text{used}[0] \setminus x_1[0])$$

$$c_1 \equiv \text{Count}(x_2 \sqcap x_1[1])$$

$$c_2 \equiv \text{Count}(x_2 \sqcap x_3 \sqcap x_1[1])$$

Grid

Consider the set of elements $\{x_1, x_2\}$ with

$$x_1 \equiv \text{at}_g[0, 1] \setminus \text{at}[0, 1]$$

$$x_2 \equiv \exists \text{key-shape}[0, 1].\exists \text{lock-shape}[0, 1].\text{locked}[0]$$

where x_1 is a role that represents the set of misplaced key paired with its respective target location, and x_2 is a concept that represent the set of keys for which a closed lock with the same shape as the key exists. The features $\Phi = \{l, k, o, t\}$ are constructed as follows.

$$l \equiv \text{Count}(\text{locked}[0])$$

$$k \equiv \text{Count}(x_1)$$

$$o \equiv \text{Nonempty}(\text{holding} \sqcap x_2)$$

$$t \equiv \text{Nonempty}(\text{holding} \sqcap x_1[0])$$

Childsnack

Consider the set of elements $\{x_1, x_2\}$ with

$$x_1 \equiv \text{served}_g[0] \setminus \text{served}[0]$$

$$x_2 \equiv \text{no-gluten-sandwich}[0]$$

where x_1 is a concept that represent the set of unserved children, and x_2 is a concept that represents the set of gluten free sandwiches. The features $\Phi = \{c_g, c_r, s_g^k, s_g^t, s_g^t\}$ are constructed as follows.

$$c_g \equiv \text{Count}(\text{allergic-gluten}[0] \sqcap x_1)$$

$$c_r \equiv \text{Count}(\text{not-allergic-gluten}[0] \sqcap x_1)$$

$$s_g^k \equiv \text{Nonempty}(\text{at-kitchen-sandwich}[0] \sqcap x_2)$$

$$s_g^t \equiv \text{Nonempty}(\text{at-kitchen-sandwich}[0] \setminus x_2)$$

$$s^t \equiv \text{Nonempty}(\text{ontray}[0])$$

Driverlog

Consider the set of elements $\{x_1, x_2, x_3, x_4\}$ with

$$x_1 \equiv (\text{at}_g[0, 1] \setminus \text{at}[0, 1])$$

$$x_2 \equiv (\text{at}[1, 0] \sqcup \text{driving}[1, 0])|_{\text{at}_g[0] \sqcap \text{driver}[0]}^{-1}$$

$$x_3 \equiv (\text{driving}[1] \sqcup (\text{at}[1, 0]|_{\text{driver}[0]}[0]))$$

$$x_4 \equiv \text{at}[0, 1] \sqcup \text{at}[1, 0] \sqcup \text{path}[0, 1]$$

where x_1 is a role that represents misplaced packages, drivers, and trucks paired with their respective target location, x_2 is a role that represents misplaced drivers paired with their current location, x_3 is a concept that represent trucks with a driver boarded and location of unboarded drivers, and x_4 is a role that represents reachable trucks and locations over unboarding, boarding, walking actions. The features $\Phi = \{p, t, d_g, d_t, b, l\}$ are constructed as follows.

$$p \equiv \text{Count}(\text{obj}[0] \sqcap x_1[0])$$

$$t \equiv \text{Count}(\text{truck}[0] \sqcap x_1[0])$$

$$d_g \equiv \text{SumRoleDist}(x_2, x_4, \text{at}_g[0, 1])$$

$$d_t \equiv \text{ConceptDist}(x_3, x_4, \text{truck}[0] \sqcap x_1[0])$$

$$b \equiv \text{Nonempty}(\text{driving}[0])$$

$$l \equiv \text{Nonempty}(\text{obj}[0] \sqcap \text{in}[0] \sqcap \text{at}_g[0])$$

Schedule

The features $\Phi = \{p_1, p_2, p_3, h, o\}$ are constructed as follows

$$p_1 \equiv \text{Count}(\text{shape}_g[0, 1] \setminus \text{shape}[0, 1])$$

$$p_2 \equiv \text{Count}(\text{surface-condition}_g[0, 1] \setminus \text{surface-condition}[0, 1])$$

$$p_3 \equiv \text{Count}(\text{painted}_g[0, 1] \setminus \text{painted}[0, 1])$$

$$h \equiv \text{Count}(\text{temperature}[0, 1]|_{\text{hot}})$$

$$o \equiv \text{Nonempty}(\text{schedule}[0] \sqcup \text{busy}[0])$$

References

Baader, F.; Calvanese, D.; McGuinness, D.; Patel-Schneider, P.; and Nardi, D. 2003. *The description logic handbook: Theory, implementation and applications*. Cambridge U.P.

Francès, G.; Bonet, B.; and Geffner, H. 2021. Learning General Policies from Small Examples Without Supervision. In *Proc. AAAI*.