Proofs of the theorems in *Search Space Expansion for Efficient Incremental Inductive Logic Programming from Streamed Data*

Theorem 1. *The set $\mathcal{G}(T') = generalise(\mathcal{C}(T), \mathcal{C}(T')) \cup \mathcal{G}(T)$ is a generalised rule space w.r.t. $\mathcal{C}(T')$.*

Proof. First, we show that all rules in the generalised rule space w.r.t. $\mathcal{C}(T')$ are in $\mathcal{G}(T')$. To do this, it suffices to show that if a rule R in the generalised rule space is not in $\mathcal{G}(T)$ then it is in the set returned by $generalise(\mathcal{C}(T), \mathcal{C}(T'))$. To do so, we show the following condition is an invariant for the main for loop: $\langle R^*, subs\rangle \in G$, where R^* is the rule with the same head as R and a body containing those literals in the body of R that have already been processed and $subs = c^+_{R^*}(T')$. Note that such a tuple can never be removed by the *Filter* function because for each element of *bls* that is not in the body of R, there must be at least one rule in $c^+_{R^*}(T')$ that does not contain it (or R could not be in the generalised rule space). Hence, it remains to show that if the condition holds at the beginning of an iteration then the condition holds for *NewG* at the end of the iteration. Consider an arbitrary iteration where the condition holds at the beginning.

Case 1: $bl \notin body(R)$. As $R \in \mathcal{G}(T')$ there must be at least one rule in $c^+_R(T)$ that does not contain *bl*. Hence, $new\_subs \neq subs$. So $\langle R^*, subs\rangle$ is added to *NewG*, so the invariant still holds.

Case 2: $bl \in body(R)$. Consider the iteration of the second for loop which corresponds to the pair $\langle R^*, subs\rangle$. Let R^*_{bl} be the rule R^* with *bl* appended to the body. Note that $new\_subs = c^+_{R^*_{bl}}(T')$. This set clearly contains $c^+_R(T')$ and so must be non-empty (as $R \in \mathcal{G}(T')$). Hence, the pair $\langle R^*_{bl}, new\_subs\rangle$ must be added to *NewG*, either in line 11 or in line 15. So the invariant still holds at the end of the iteration.

Hence, at the end of the execution, $\langle R, c^+_R(T')\rangle \in G$, and so R is in the set returned by *generalise*.

It remains to show that every rule R in $\mathcal{G}(T')$ is in the generalised rule space w.r.t. $\mathcal{C}(T')$.

Case 1: $R \in \mathcal{G}(T)$. Then $c^+_R(T) \neq \emptyset$ and there is no rule $R' \in S_M$ s.t. R is a strict sub-rule of R' and $c^+_R(T) = c^+_{R'}(T)$. Hence $c^+_R(T') \neq \emptyset$ and there is no rule $R' \in S_M$ s.t. R is a strict sub-rule of R' and $c^+_R(T') = c^+_{R'}(T')$. So, R must be in $\mathcal{G}(T')$.

Case 2: $R \in generalise(\mathcal{C}(T), \mathcal{C}(T'))$. Then there must be a pair $\langle R, subs\rangle \in G$ at the end of the *generalise* execution. Assume for contradiction that $c^+_R(T') = \emptyset$. Then for each $R' \in \mathcal{C}^+(T)$ s.t. $head(R) = head(R')$, at least one element of *bls* must be in $body(R)$ and not in $body(R')$. Hence, there must be an iteration where *new_subs* becomes empty, and no pair $\langle R, S\rangle$ is added to *NewG*. This contradicts the fact that $\langle R, subs\rangle \in G$ at the end. It remains to show that there is no $R' \in S_M$ s.t. R is a strict sub-rule of R' and $c^+_R(T) = c^+_{R'}(T)$. If this were the case then each body literal that occurs in R' but not in R would have to occur in every element of $c^+_R(T)$. But if this were the case, then the rule would be removed by the *Filter* at some point in the execution. Hence, R must be in $\mathcal{G}(T')$. □

Theorem 2. *Let $\mathcal{O}(T') = \{\langle R, \mathcal{O}_{R,T'}\rangle \mid R \in \mathcal{U}(T, T')\} \cup \{\langle R, \mathcal{O}_{R,T}\rangle \mid R \in \mathcal{G}(T') \backslash \mathcal{U}(T, T')\}$. $\langle \mathcal{C}(T'), \mathcal{G}(T'), \mathcal{O}(T')\rangle$ is a valid state after solving T'.*

Proof. As the sets $\mathcal{U}(T, T')$ and $\mathcal{G}(T') \backslash \mathcal{U}(T, T')$ form a partition of the set $\mathcal{G}(T')$, each rule in $\mathcal{G}(T')$ is clearly represented by exactly one pair in $\mathcal{O}(T')$. It therefore remains to show that the second element of this pair is a valid optimisation of R w.r.t. T'.

Case 1: $R \in \mathcal{U}(T, T')$. In this case the second element, $\mathcal{O}_{R,T'}$ is defined according to Definition 4 of the main paper (using $\mathcal{G}(T')$ and $\mathcal{C}(T')$); hence, due to the correctness results proved in [Law *et al.*, 2020] it must be a valid optimisation of R w.r.t. T'.

Case 2: $R \notin \mathcal{U}(T, T')$. In this case, we must show that $\mathcal{O}_{R,T}$ is a valid optimisation of R w.r.t. T. Assume for contradiction that it is not. Then either there is a rule in $\mathcal{O}_{R,T}$ that violates one of the conditions in (1) of Definition 4, or an additional rule can be added without violating (1). As R is not in $\mathcal{U}(T, T')$, for each rule in $r \in \mathcal{O}_{R,T}$, $v(r, T) = v(r, T')$. Hence, as for any other rule in $r' \in S_M$, $v(r', T) \subseteq v(r', T')$, this cannot be the case. Contradiction! □

Theorem 3. *For any task T and valid state, $\langle \mathcal{C}(T)$, $\mathcal{G}(T)$, $\mathcal{O}(T)\rangle$, after solving T, $\bigcup_{\langle R,S\rangle \in \mathcal{O}(T)} R$ is OPT-sufficient.*

Proof. Assume for contradiction that $O = \bigcup\limits_{\langle R,S\rangle \in \mathcal{O}(T)} R$ is not OPT-sufficient. Then T must be satisfiable. Let H^* be an optimal solution of T. There must be a rule in $h \in H^*$ such that $c^+_h(T) \neq \emptyset$ and there is no rule $h' \in O$ s.t. $|h| = |h'|$, $c^+_h(T) = c^+_{h'}(T)$ and $v(h', T) \subseteq v(h, T)$ – otherwise each rule R could be replaced with its corresponding rule R', and the resulting hypothesis would be an optimal solution.

As $\mathcal{G}(T)$ is a generalised rule space, there must be a rule $h^g \in \mathcal{G}(T)$ such that $c^+_h(T) = c^+_{h^g}(T)$ and h is a sub-rule of h^g. But if this were the case, as $\mathcal{O}(R,T) \subset O$, $\mathcal{O}(R,T)$ must not be a valid optimisation of R w.r.t. T, as h can be added to it without breaking (1) of Definition 4. Contradiction! $\square$

References

[Law *et al.*, 2020] Mark Law, Alessandra Russo, Elisa Bertino, Krysia Broda, and Jorge Lobo. FastLAS: Scalable inductive logic programming incorporating domain-specific optimisation criteria. In *AAAI*. Association for the Advancement of Artificial Intelligence, 2020.