On the Expressive Power of Planning Formalisms in Conjunction with LTL

Songtuan Lin   Pascal Bercher

School of Computing
College of Engineering & Computer Science
The Australian National University

June 3, 2022

Australian
National
University

Motivation

We want to study the expressiveness of both hierarchical and non-hierarchical planning frameworks in conjunction with Linear Temporal Logic (LTL) in order to know:

- Whether LTL can *actually* improve the expressive power of a planning framework
- What is the upper bound of the expressiveness of a planning framework when combining with LTL
- Which problem class can be modeled by a certain planning framework with LTL

Approach

- Expressiveness – The class of *formal languages* that can be expressed
- For the purpose of studying the expressiveness of a planning framework incorporating LTL
    - We view the solution set of a planning problem in the target formalism as a formal language
    - We compare the language of a planning problem with other languages

LTL

- The syntax of LTL:

$$\varphi = \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathtt{U}\, \varphi_2$$

- The semantics of LTL: Given a state sequence $\pi = \langle s_1 \cdots \rangle$, we define $\pi[\![i]\!] = \langle s_i \cdots \rangle$
  - $\pi[\![i]\!] \models \top$
  - $\pi[\![i]\!] \models p$ *iff* $p \in s_i$
  - $\pi[\![i]\!] \models \neg\varphi$ *iff* $\pi[\![i]\!] \not\models \varphi$
  - $\pi[\![i]\!] \models \bigcirc\varphi$ *iff* $\pi[\![i+1]\!] \models \varphi$
  - $\pi[\![i]\!] \models \varphi_1 \wedge \varphi_2$ *iff* $\pi[\![i]\!] \models \varphi_1 \wedge \pi[\![i]\!] \models \varphi_2$
  - $\pi[\![i]\!] \models \varphi_1 \,\mathtt{U}\, \varphi_2$ *iff* there exists a $j \geq i$ such that $\pi[\![j]\!] \models \varphi_2$ and $\pi[\![k]\!] \models \varphi_1$ for all $i \leq k < j$.
- The semantics is defined over an *infinite* state sequence

Finite LTL (f-LTL)

- The syntax of f-LTL is identical to that of standard LTL
- The semantics of f-LTL is defined over a state sequence $\pi = \langle s_1 \cdots s_n \rangle$ of *finite* length
  - $\pi_i \models \bigcirc\varphi$ *iff* $i < n$ and $\pi_{i+1} \models \varphi$
  - $\pi_i \models \varphi_1 \,\mathtt{U}\, \varphi_2$ *iff* there exists a j with $i \leq j \leq n$ such that $\pi_j \models \varphi_2$, and for each $i \leq k < j$, $\pi_k \models \varphi_1$
- f-LTL can express *the end of a state sequence* $\odot$:

$$\odot = \bigcirc(\neg\top)$$

i.e., $\pi[\![i]\!] \models \odot$ *iff* $i = n$ (s_i is the last in π)

Expressiveness of Classical Planning Framework with LTL

Planning Framework: $\mathcal{STRIPS}$

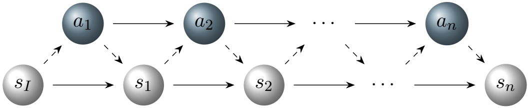
- A $\mathcal{STRIPS}$ planning problem $\mathcal{P}$ is a tuple:

$$\mathcal{P} = (\underbrace{\mathcal{F}}_{\substack{\text{a set of}\\ \text{propositions}}}, \underbrace{\mathcal{A}}_{\substack{\text{a set of}\\ \text{actions}}}, \underbrace{\delta}_{\substack{\text{a function:}\\ \mathcal{A}\to 2^{\mathcal{F}}\times 2^{\mathcal{F}}\times 2^{\mathcal{F}}}}, \underbrace{s_I}_{\substack{\text{initial}\\ \text{state}}}, \underbrace{g}_{\substack{\text{goal}\\ g\subseteq\mathcal{F}}})$$

- δ maps each action to its precondition, positive effects, and negative effects so that we can view each action as a *symbol*:

$$\delta(a) = (prec(a), eff^{+}(a), eff^{-}(a))$$

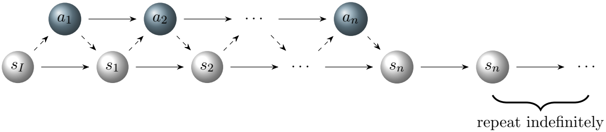
- A solution is an action sequence $\overline{a} = \langle a_1 \cdots a_n \rangle$ such that



  - $prec(a_i) \subseteq s_{i-1}$
  - $g \subseteq s_n$
  - $s_i = (s_{i-1} \setminus eff^{-}(a_i)) \cup eff^{+}(a_i)$

Planning Framework: $\mathcal{STRIPS}$-$\mathcal{L}$ and $\mathcal{STRIPS}$-$\mathcal{FL}$

- A $\mathcal{STRIPS}$-$\mathcal{L}$ planning problem is like a $\mathcal{STRIPS}$ planning problem except that g is an LTL formula



  - A solution is an action sequence $\overline{a} = \langle a_1 \cdots a_n \rangle$ such that $\tilde{\pi} = \langle s_I \ s_1 \cdots s_n \ s_n \cdots \rangle \models g$ where $\pi = \langle s_I \ s_1 \cdots s_n \rangle$ is obtained by applying $\overline{a}$ in s_I
  - **Note** that although the state sequence is extended to infinite, the solution is still **finite**
- A $\mathcal{STRIPS}$-$\mathcal{FL}$ planning problem is like a $\mathcal{STRIPS}$ planning problem except that g is an f-LTL formula
  - A solution is an action sequence $\overline{a}$ which leads to a state sequence π satisfying g
  - Note that π does *not* need to be extended to infinite

Languages of Planning Problems/Formalisms

- The language of a planning problem $\mathcal{P}$ in $\mathcal{STRIPS}$, $\mathcal{STRIPS}$-$\mathcal{L}$, or $\mathcal{STRIPS}$-$\mathcal{FL}$ formalism:

$$\mathcal{L}(\mathcal{P}) = \{\overline{a} \mid \overline{a} \text{ is a solution to } \mathcal{P}\}$$

- The class of languages of a planning formalism X with X being $\mathcal{STRIPS}$, $\mathcal{STRIPS}$-$\mathcal{L}$, or $\mathcal{STRIPS}$-$\mathcal{FL}$:

$$\mathcal{L}_X = \{\mathcal{L}(\mathcal{P}) \mid \mathcal{P} \text{ is a planning problem in the formalism } X\}$$

Theoretical Results

Theorem

$\mathcal{L}_{\mathcal{STRIPS}} \subsetneq \mathcal{L}_{\mathcal{STRIPS}\text{-}\mathcal{L}} \subsetneq \mathcal{L}_{\mathcal{STRIPS}\text{-}\mathcal{FL}} = \mathcal{SF} \subsetneq \mathcal{REG}$
where $\mathcal{SF}$ refers to the class of star-free languages, which is a strict subset of regular languages ($\mathcal{REG}$)

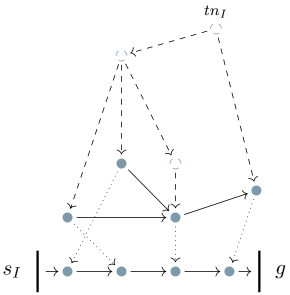
Proof Ideas

- The star-free language $\{\langle a\ a \rangle\}$ cannot be expressed by the $\mathcal{STRIPS}$ or $\mathcal{STRIPS}$-$\mathcal{L}$
- $\{\langle a\ a \rangle\}$ can be expressed by the $\mathcal{STRIPS}$-$\mathcal{FL}$ formalism

Expressiveness of Hierarchical Planning Framework with LTL

Planning Framework: $\mathcal{HTN}$ Planning

An $\mathcal{HTN}$ planning problem $\mathcal{P}$ is a tuple $(\mathcal{D}, c_I, s_I, g)$ where $\mathcal{D} = (\mathcal{F}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \delta)$ is the domain



- $\mathcal{C}$ is a set of compound tasks
- $\mathcal{M}$ is a set of methods
- $c_I \in \mathcal{C}$ is the initial compound task
- A compound task is decomposed into a task network by a method
- A task network is a partial order set of actions and compound tasks
- A solution is a task network tn consisting of actions
    - tn is obtained from c_I
    - tn has an executable linearization in s_I
    - g is satisfied

Planning Framework: Variants of $\mathcal{HTN}$

$\mathcal{TIHTN}$ – $\mathcal{HTN}$ planning with task insertions

- Actions can be inserted to task networks
- A solution is a task network obtained by decomposition and task insertions

$\mathcal{TOHTN}$ – a special case of $\mathcal{HTN}$ planning

- Every method is totally ordered

$(\mathcal{TI})\mathcal{HTN}$-$\mathcal{L}$/$(\mathcal{TI})\mathcal{HTN}$-$\mathcal{FL}$ – Combination with LTL/f-LTL

- g is expressed in terms of an LTL/f-LTL formula

Languages of Planning Problems/Formalisms

- The language of a planning problem $\mathcal{P}$ in the formalism $(\mathcal{TI})\mathcal{HTN}$, $(\mathcal{TI})\mathcal{HTN}$-$\mathcal{L}$, or $(\mathcal{TI})\mathcal{HTN}$-$\mathcal{FL}$ is

$$\mathcal{L}(\mathcal{P}) = \left\{ \pi \;\middle|\; \begin{array}{l} \pi \text{ is an executable linearization of } tn, \\ tn \text{ is a solution to } \mathcal{P} \end{array} \right\}$$

- The class of languages of a hierarchical planning formalism X with X being $(\mathcal{TI})\mathcal{HTN}$, $(\mathcal{TI})\mathcal{HTN}$-$\mathcal{L}$, or $(\mathcal{TI})\mathcal{HTN}$-$\mathcal{FL}$ is

$$\mathcal{L}_X = \{\mathcal{L}(\mathcal{P}) \mid \mathcal{P} \text{ is a planning problem in the formalism } X\}$$

Theoretical Results: $\mathcal{TIHTN}$

Theorem

$\mathcal{L}_{\mathcal{TIHTN}} \subsetneq \mathcal{L}_{\mathcal{TIHTN}\text{-}\mathcal{L}} \subsetneq \mathcal{L}_{\mathcal{TIHTN}\text{-}\mathcal{FL}} = \mathcal{SF}$

Proof Ideas

- The language of a hierarchical planning problem can be viewed as the intersection of the language of its hierarchical part and that of its non-hierarchical part

Theoretical Results: $\mathcal{TOHTN}$

Theorem

$\mathcal{L}_{\mathcal{TOHTN}} = \mathcal{L}_{\mathcal{TOHTN}\text{-}\mathcal{L}} = \mathcal{L}_{\mathcal{TOHTN}\text{-}\mathcal{FL}} = \mathcal{CFL}$ where $\mathcal{CFL}$ refers to the class of context-free languages

Proof Ideas

- The language of the hierarchical part is context-free
- The language of the non-hierarchical part is regular
- The intersection of a context-free language and a regular language is still a context-free language

Theoretical Results: $\mathcal{HTN}$

Theorem

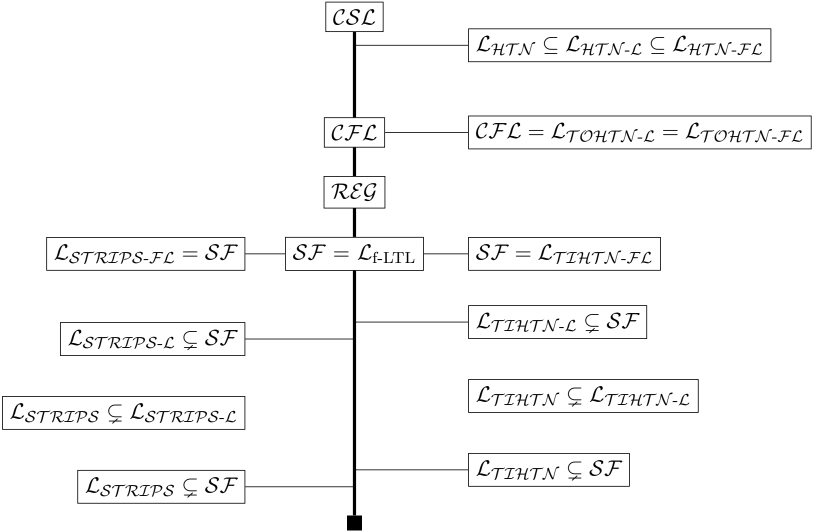
$\mathcal{CFL} \subsetneq \mathcal{L}_{\mathcal{HTN}} \subseteq \mathcal{L}_{\mathcal{HTN}\text{-}\mathcal{L}} \subseteq \mathcal{L}_{\mathcal{HTN}\text{-}\mathcal{FL}} \subseteq \mathcal{CSL}$ where $\mathcal{CSL}$ refers to the class of context-sensitive languages

Proof Ideas

- The intersection of a context-sensitive language and a regular language is still a context-sensitive language

Conclusion

Conclusion


$\mathcal{CSL}$
$\mathcal{L}_{\mathcal{HTN}} \subseteq \mathcal{L}_{\mathcal{HTN}\text{-}\mathcal{L}} \subseteq \mathcal{L}_{\mathcal{HTN}\text{-}\mathcal{FL}}$
$\mathcal{CFL}$
$\mathcal{CFL} = \mathcal{L}_{\mathcal{TOHTN}\text{-}\mathcal{L}} = \mathcal{L}_{\mathcal{TOHTN}\text{-}\mathcal{FL}}$
$\mathcal{REG}$
$\mathcal{L}_{\mathcal{STRIPS}\text{-}\mathcal{FL}} = \mathcal{SF}$
$\mathcal{SF} = \mathcal{L}_{\text{f-LTL}}$
$\mathcal{SF} = \mathcal{L}_{\mathcal{TIHTN}\text{-}\mathcal{FL}}$
$\mathcal{L}_{\mathcal{TIHTN}\text{-}\mathcal{L}} \subsetneq \mathcal{SF}$
$\mathcal{L}_{\mathcal{STRIPS}\text{-}\mathcal{L}} \subsetneq \mathcal{SF}$
$\mathcal{L}_{\mathcal{TIHTN}} \subsetneq \mathcal{L}_{\mathcal{TIHTN}\text{-}\mathcal{L}}$
$\mathcal{L}_{\mathcal{STRIPS}} \subsetneq \mathcal{L}_{\mathcal{STRIPS}\text{-}\mathcal{L}}$
$\mathcal{L}_{\mathcal{TIHTN}} \subsetneq \mathcal{SF}$
$\mathcal{L}_{\mathcal{STRIPS}} \subsetneq \mathcal{SF}$