Program Guided Agent

Shoa-Hua Sun et al. (2019)

2020.8.9

Haneul Choi / caelum02@snu.ac.kr

Motivation

- **Learning from scratch is inefficient**

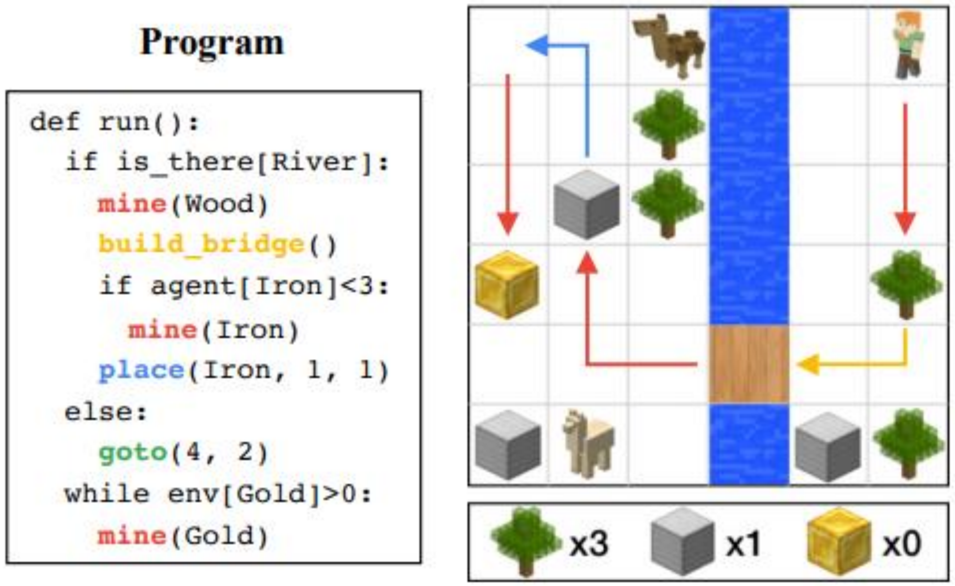
  Using program as an instruction

- **Hierarchical Approach**

  learning to deal with subtasks

- **Multitask learning**

  learn generalizable knowledge from multiple tasks



Similar Approaches

- **Expert demonstrations**

  Learning from video / expert trajectories

- **Natural language instructions**

  wide range of applications; but ambiguous

- **Hierarchical approaches**

  provide set of symbolically represented subtasks

  -> not function of states

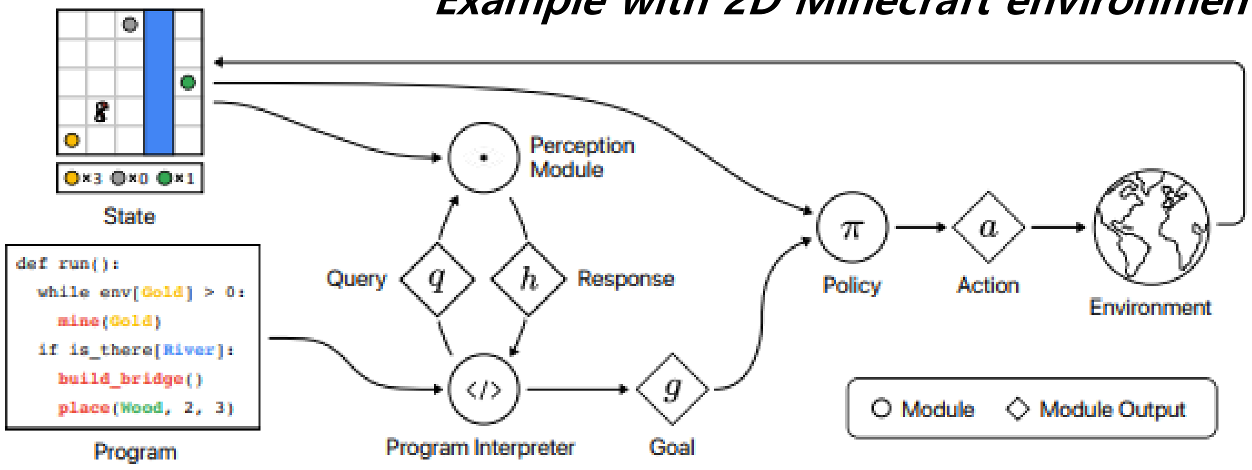
Utilize ***program instructions***

-> framework must…

- ***Comprehend program well***
- ***Perceive and interact with Env well***

Program guided agent

Example with 2D Minecraft environment



Modular architecture (instead of End-to-End model)

[Component Modules]

1_ **program interpreter** : Execute program (***query*** *perception module and* ***instruct*** *policy with subtasks*)

2_ **perception modules** : deal with queries (*e.g.* `is_there[River] / env[gold] > 0`)

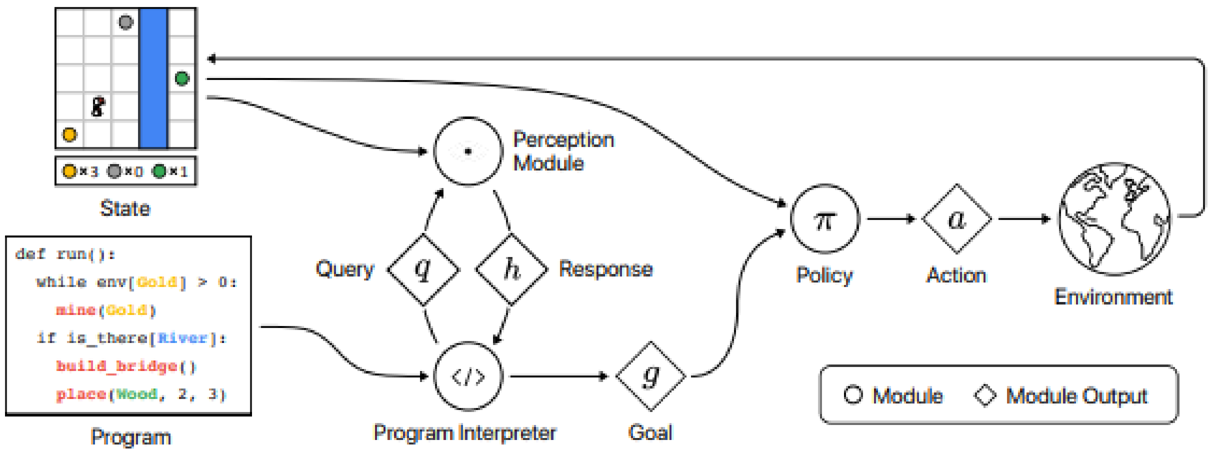
3_ **policy (multi task policy)** : solve subtasks (*e.g.* `mine(Gold) / build_bridge() ..`)

Program Interpreter

- A rule-based algorithm; ***Not a model***
- Defined based on **domain-specific language (DSL)**

$$
\begin{aligned}
\text{Program } p &:= \text{def run() : } s \\
\text{Statement } s &:= \text{while}(c) : (s) \mid b \mid \text{loop}(i) : (s) \\
&\mid \text{if}(c) : (s) \mid \text{elseif}(c) : (s) \mid \text{else} : (s) \\
\text{Item } t &:= \text{Gold} \mid \text{Wood} \mid \text{Iron} \\
\text{Terrain } u &:= \text{Bridge} \mid \text{River} \mid \text{Merchant} \mid \text{Wall} \mid \text{Flat} \\
\text{Operators } o &:= > \ \geq\ ==\ <\ \leq \\
\text{Numbers } i &:= \text{A positive integer or zero} \\
\text{Perception } h &:= \text{agent[t]} \mid \text{env[t]} \mid \text{is\_there[t]} \mid \text{is\_there[u]} \\
\text{Behavior } b &:= \text{mine(t)} \mid \text{goto(i, i)} \\
&\mid \text{place(t, i, i)} \mid \text{build\_bridge()} \mid \text{sell(t)} \\
\text{Conditions } c &:= h[t]\ o\ i \mid h[u]\ o\ i
\end{aligned}
$$

<- **perception / action primitives** can be defined by user



- Composed of :
  - **Subtasks, Perceptions, Control Flows**

Perception Module

- Receive query and state observation and predict response

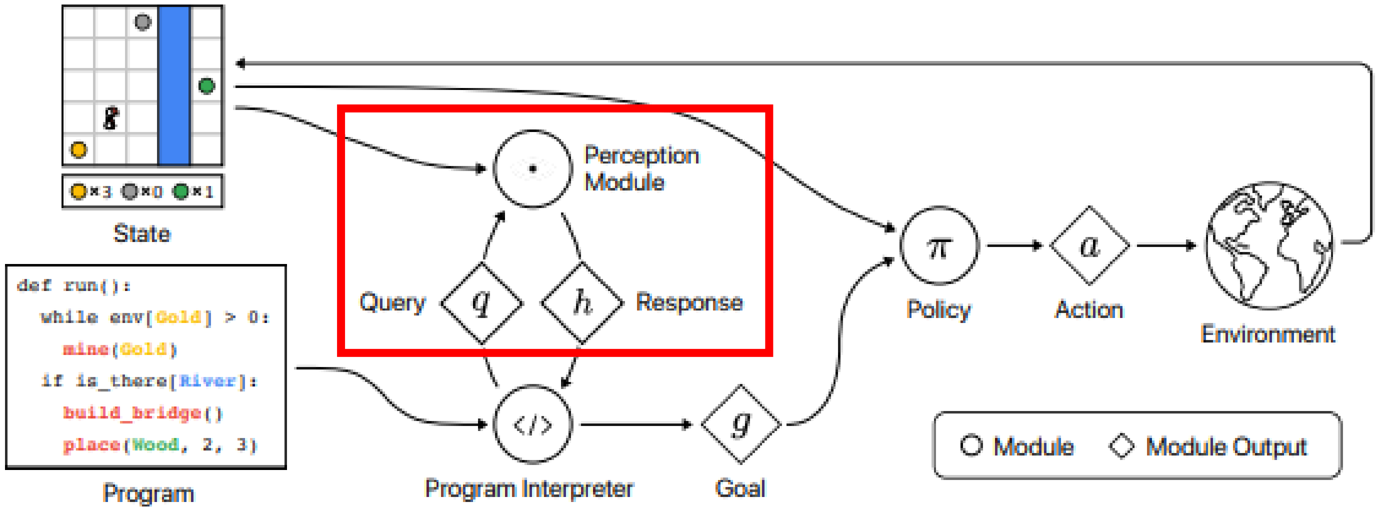
  $h = \Phi(q, s)$; Φ is perception module, h is response

- **Pretrained;** Supervised-Learning method

  With cross-entropy loss

- Queries are fed after encoding

  program tokens



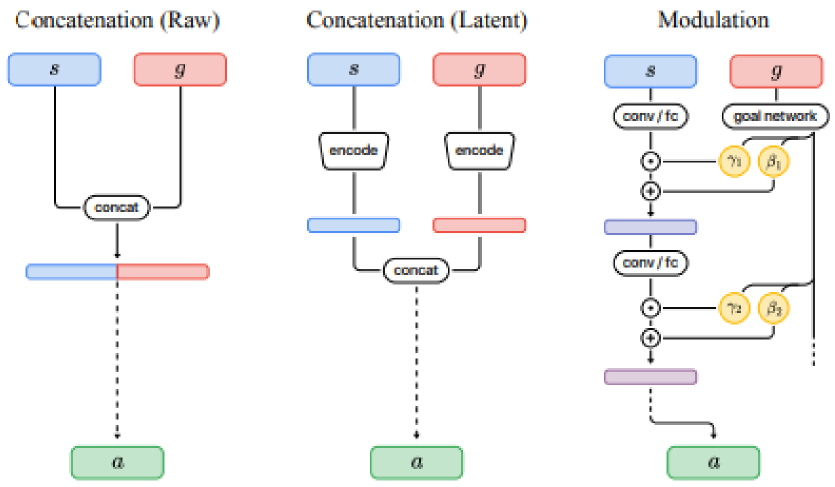
Policy Module

- **Multitask policy**
  - **perform multiple subtasks** by taking low-level actions (*e.g.* `moveUp, moveLeft`)
  - goal is instructed by interpreter, and **network take encoded goal as input**

    i.e. $a \sim \boldsymbol{\pi}(s, g)$
- **A2C (Advantage Actor-Critic) algorithm** used with entropy term
- **Modulation mechanism** used in receiving goal

  Affine transforming state observation rather than concate

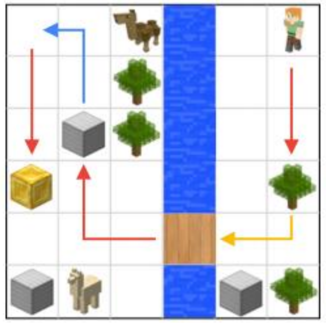
  nating



Experiments

1. Can ***program guided agent*** learn well?
2. ***Effectiveness of modular*** architecture (vs. end-to-end models)
3. Analysis on the performance of end-to-end models
4. Is ***modulation mechanism*** more efficient to learn multitasking?

[environment]



[end-to-end learning models]

- LSTM, Transformers, Tree-RNN with A2C
- Tree-RNN?

  : Rnns for tree-structured inputs

  : program -> tree structure

[training sets]

- 4500 programs sampled using DSL
  - ***Test*** (4000 programs), ***train*** (500 programs)

\+ ***Test-complex***

- twice longer programs, more condition branches
- check generalization to more complex tasks

- Reward +1 when instruction achieved, otherwise 0

(instruction i.e. entire program or natural language instruction)

Results

- Can ***program guided agent*** learn well?
- ***Effectiveness of modular*** architecture (vs. end-to-end models)

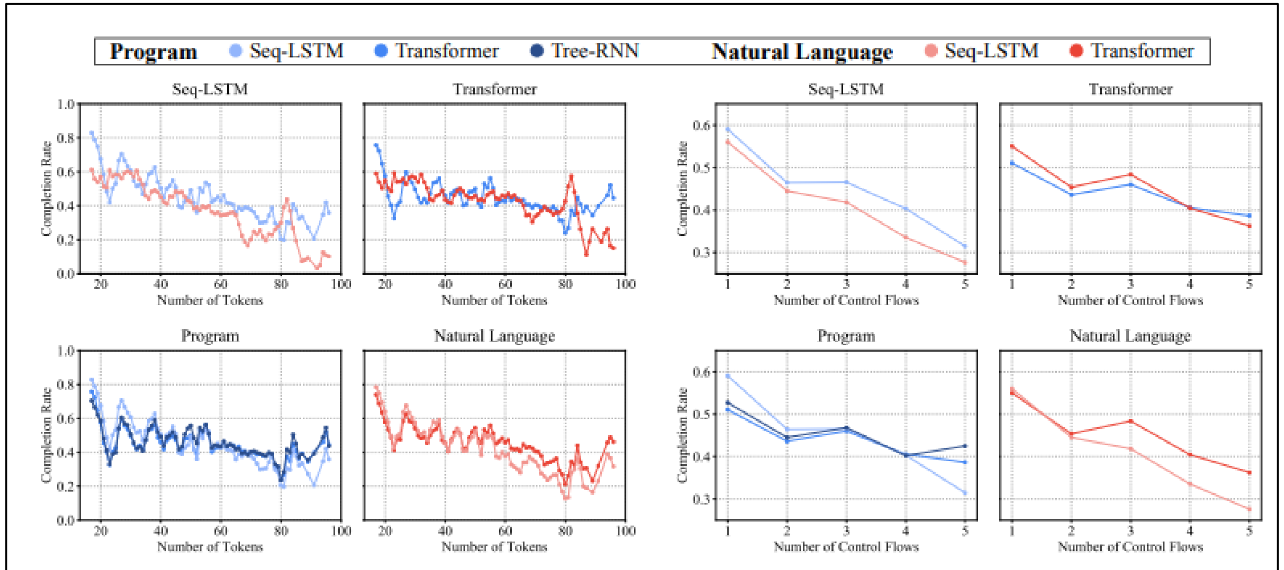
Instruction		**Natural language descriptions**		**Programs**				
Method		Seq-LSTM	Transformer	Seq-LSTM	Tree-RNN	Transformer	Ours (concat)	Ours
Dataset	test	54.9±1.8%	52.5±2.6%	56.7±1.9%	50.1±1.2%	49.4±1.6%	88.6±0.8%	94.0±0.5%
	test-complex	32.4±4.9%	38.2±2.6%	38.8±1.2%	42.2±2.4%	40.9±1.5%	85.2±0.8%	91.8±0.2%
Generalization gap		40.9%	27.2%	31.6%	15.8%	17.2%	3.8%	2.3%

(average task completion rate ± std.)

- ***Program guided agent*** **does learns** to solve multiple tasks
- ***Program guided agent*** **does generalize well** compared to **end-to-end models**
  - *- Smaller performance drop in test-complex*
- Transformer generalize better than LSTM
- Tree-RNN generalizes best among end-to-end models

Results

- Analysis on **end-to-end models**



(a) Instruction Length
(b) Instruction Diversity

Instruction Diversity : number of control flows

[Instruction length]

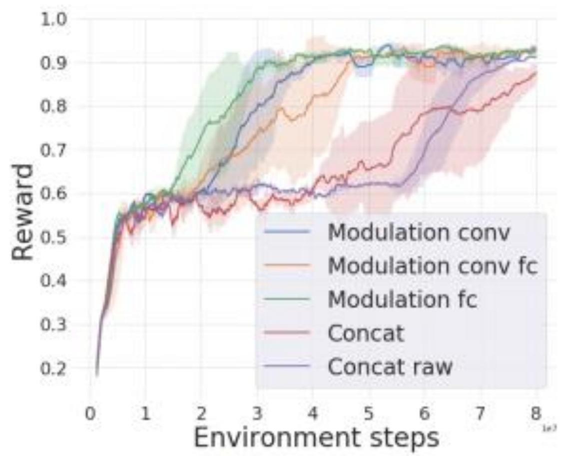
- Transformer – performance is similar across task and complex-task
- Tree-RNN achieves best performance on programs

[Instruction diversity]

- Transformer – more robust with diversity
  -> Transformer learns semantics well
- Tree-RNN achieves best performance on programs

Results

- Modulation mechanism for learning multi-task policy



(b) Training curves

Instruction Method		Natural language descriptions Seq-LSTM	Transformer	Programs Seq-LSTM	Tree-RNN	Transformer	Ours (concat)	Ours
Dataset	test	54.9±1.8%	52.5±2.6%	56.7±1.9%	50.1±1.2%	49.4±1.6%	88.6±0.8%	94.0±0.5%
	test-complex	32.4±4.9%	38.2±2.6%	38.8±1.2%	42.2±2.4%	40.9±1.5%	85.2±0.8%	91.8±0.2%
Generalization gap		40.9%	27.2%	31.6%	15.8%	17.2%	3.8%	2.3%

- *Modulation mechanism* is more **sample efficient**
- Using *Modulation mechanism* help achieving **better performance in learning multitask policy**

Conclusion

- Modular framework ***Program guided agent*** learn program based instructions well
- ***Modulation mechanism*** is efficient; sample efficiency, task completion
- Can ***generalize*** to more complex tasks <u>without additional learning</u>

  : Zero shot learning !