BABYAI AND RESTRAINING BOLTS

Reasoning Agents – Prof. De Giacomo

Who we are

github.com/francesco-AI/ReasoningAgent2020

FRANCESCO CASSINI

7857771

BOTA DUISENBAY

1849680

DAVIDE GIORDANO

1852364

FRANCESCO VINCELLI

1727959

Index

1

Presentation of BabyAI paper

2

NLP to LDLf

3

Imitation Learning theory

4

Implementation

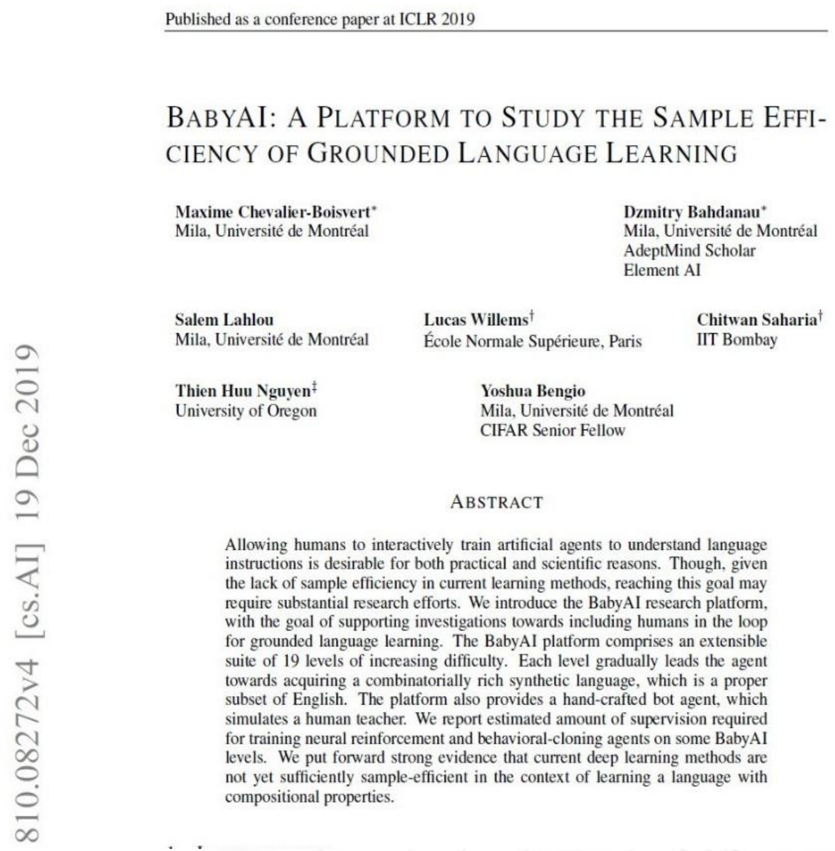
1. Presentation of BabyAI paper

An introduction to the paper from which we took inspiration

Introduction

BabyAI is a paper written in 2019 that aims to teach a bot to acquire a combinatorially rich synthetic language;

BabyAI is also a research platform towards including humans in the loop for grounded language learning.



Published as a conference paper at ICLR 2019

BABYAI: A PLATFORM TO STUDY THE SAMPLE EFFICIENCY OF GROUNDED LANGUAGE LEARNING

Maxime Chevalier-Boisvert*
Mila, Université de Montréal

Dzmitry Bahdanau*
Mila, Université de Montréal
AdeptMind Scholar
Element AI

Salem Lahlou
Mila, Université de Montréal

Lucas Willems†
École Normale Supérieure, Paris

Chitwan Saharia†
IIT Bombay

Thien Huu Nguyen‡
University of Oregon

Yoshua Bengio
Mila, Université de Montréal
CIFAR Senior Fellow

ABSTRACT

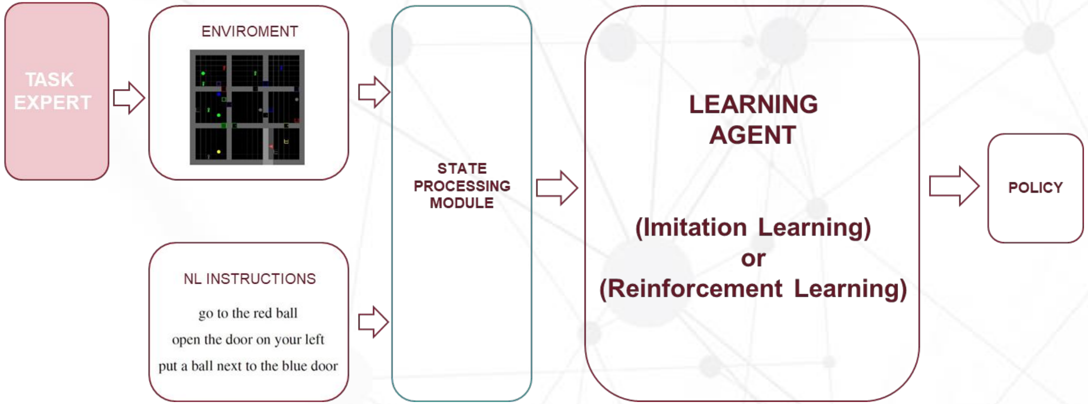
Allowing humans to interactively train artificial agents to understand language instructions is desirable for both practical and scientific reasons. Though, given the lack of sample efficiency in current learning methods, reaching this goal may require substantial research efforts. We introduce the BabyAI research platform, with the goal of supporting investigations towards including humans in the loop for grounded language learning. The BabyAI platform comprises an extensible suite of 19 levels of increasing difficulty. Each level gradually leads the agent towards acquiring a combinatorially rich synthetic language, which is a proper subset of English. The platform also provides a hand-crafted bot agent, which simulates a human teacher. We report estimated amount of supervision required for training neural reinforcement and behavioral-cloning agents on some BabyAI levels. We put forward strong evidence that current deep learning methods are not yet sufficiently sample-efficient in the context of learning a language with compositional properties.

BabyAI – Previous works

To understand BabyAI network we have to cite some previous research in the field of grounded language learning.

“Gated-Attention Architectures for Task-Oriented Language Grounding”

[2018: Chaplot, Sathyendra et others]

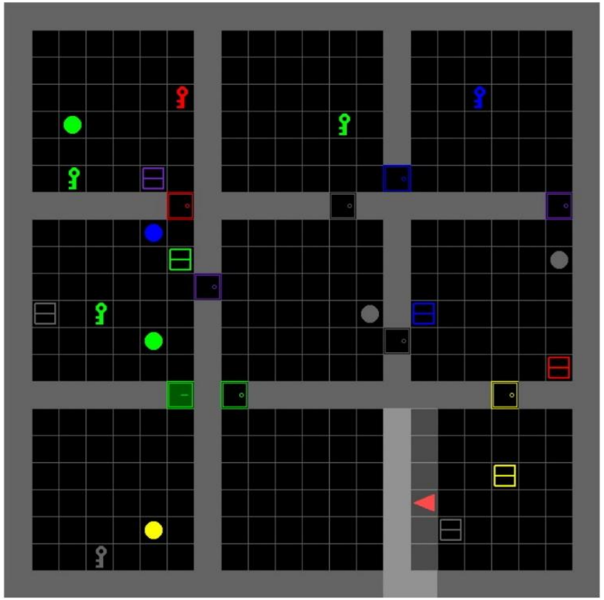


BabyAI – Platform Description

The elements that characterize BabyAI platform.

Minigrid

The MiniGrid environment is fast and this makes experimentation quicker and more accessible.



BabyAI – Platform Description

The elements that characterize BabyAI platform.

Baby Language

The instructions are in a simplified subset of English and it's possible combine more instructions together.

⟨Sent⟩	⊨	⟨Sent1⟩ ∣ ⟨Sent1⟩ ',' then ⟨Sent1⟩
⟨Sent1⟩	⊨	⟨Clause⟩ ∣ ⟨Clause⟩ and ⟨Clause⟩
⟨Clause⟩	⊨	go to ⟨Descr⟩ ∣ pick up ⟨DescrN put ⟨DescrNotDoor⟩ next to ⟨Desc
⟨DescrDoor⟩	⊨	⟨Article⟩ ⟨Color⟩ door ⟨LocSpec⟩
⟨DescrBall⟩	⊨	⟨Article⟩ ⟨Color⟩ ball ⟨LocSpec⟩
⟨DescrBox⟩	⊨	⟨Article⟩ ⟨Color⟩ box ⟨LocSpec⟩
⟨DescrKey⟩	⊨	⟨Article⟩ ⟨Color⟩ key ⟨LocSpec⟩
⟨Descr⟩	⊨	⟨DescrDoor⟩ ∣ ⟨DescrBall⟩ ∣ ⟨De
⟨DescrNotDoor⟩	⊨	⟨DescrBall⟩ ∣ ⟨DescrBox⟩ ∣ ⟨De
⟨LocSpec⟩	⊨	ϵ ∣ on your left ∣ on your right ∣
⟨Color⟩	⊨	ϵ ∣ red ∣ green ∣ blue ∣ purple
⟨Article⟩	⊨	the ∣ a

BabyAI – Platform Description

The elements that characterize BabyAI platform.

Levels and Curriculum learning

To investigate on curriculum learning, the environment is composed by 19 levels in which the difficulty are gradually increased.

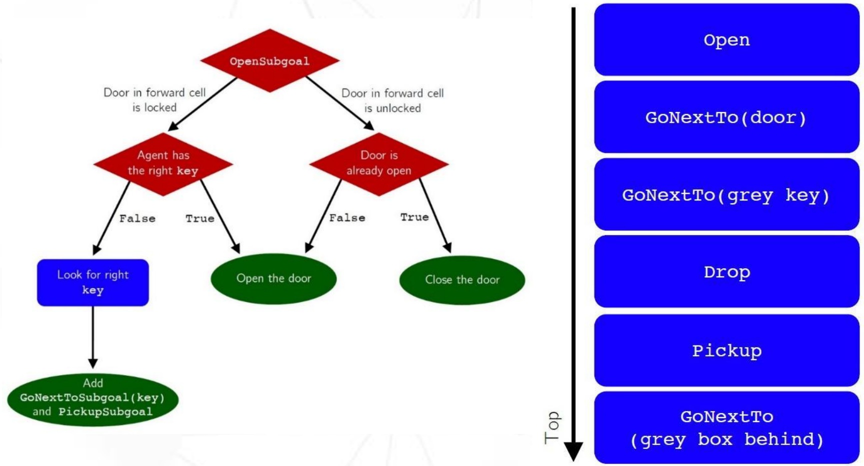
	ROOM	DISTR-BOX	DISTR	MAZE	UNBLOCK	UNLOCK	IMP-UNLOCK	GOTO	OPEN	PICKUP	PUT	LOC	SEQ
GoToObj	x												
GoToRedBallGrey	x	x											
GoToRedBall	x	x	x										
GoToLocal	x	x	x					x					
PutNextLocal	x	x	x								x		
PickupLoc	x	x	x							x		x	
GoToObjMaze	x			x									
GoTo	x	x	x	x				x					
Pickup	x	x	x	x						x			
UnblockPickup	x	x	x	x	x					x			
Open	x	x	x	x					x				
Unlock	x	x	x	x		x			x				
PutNext	x	x	x	x							x		
Synth	x	x	x	x	x	x		x	x	x	x		
SynthLoc	x	x	x	x	x	x		x	x	x	x	x	
GoToSeq	x	x	x	x				x					x
SynthSeq	x	x	x	x	x	x		x	x	x	x	x	x
GoToImpUnlock	x	x	x	x			x	x					
BossLevel	x	x	x	x	x	x	x	x	x	x	x	x	x

BabyAI – Platform Description

The elements that characterize BabyAI platform.

The Bot

The Bot plays the role of the human in the loop.



2. NLP to LDLf

Why ?

Mapping from free unbound natural language to its machine understandable representation is required:

1. Formalization to avoid ambiguity

2. Natural to untrained domain experts:
   a. Wider application
   b. No more training requirement

"After the button is pressed, the light will turn red until the elevator arrives at the floor and the doors open."

$\Updownarrow$

$p \rightarrow X (q \, U (s \wedge v))$

p - button being pressed

q - the light turning red

s - the elevator arriving

v- the doors opening

What? LTLf vs LDLf

Linear Dynamic Logic on finite traces (LDLf) - more expressive extension of LTLf, that results from addition of Propositional Dynamic Logic syntax

$$\varphi ::= \phi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \circ\varphi \mid \bullet\varphi \mid \Diamond\varphi \mid \Box\varphi \mid \varphi_1 \mathcal{U} \varphi_2$$

where ϕ is a propositional formula over $\mathcal{P}$, $\circ$ is the *next* operator, $\bullet$ is *weak next*, i.e., $\bullet\varphi$ is an abbreviation for $\neg\circ\neg\varphi$[1], $\Diamond$ is *eventually*, $\Box$ is *always*, and $\mathcal{U}$ is *until*.

$$\begin{aligned} \varphi &::= \phi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \langle\rho\rangle\varphi \mid [\rho]\varphi \\ \rho &::= \phi \mid \varphi? \mid \rho_1 + \rho_2 \mid \rho_1 ; \rho_2 \mid \rho^* \end{aligned}$$

where ϕ is a propositional formula over $\mathcal{P}$; ρ denotes path expressions, which are regular expressions over propositional formulas ϕ with the addition of the test construct $\varphi?$ typical of PDL; and φ stands for LDL_f formulas built by applying boolean connectives and the modal connectives $\langle\rho\rangle\varphi$ and $[\rho]\varphi$. In fact $[\rho]\varphi \equiv \neg\langle\rho\rangle\neg\varphi$.

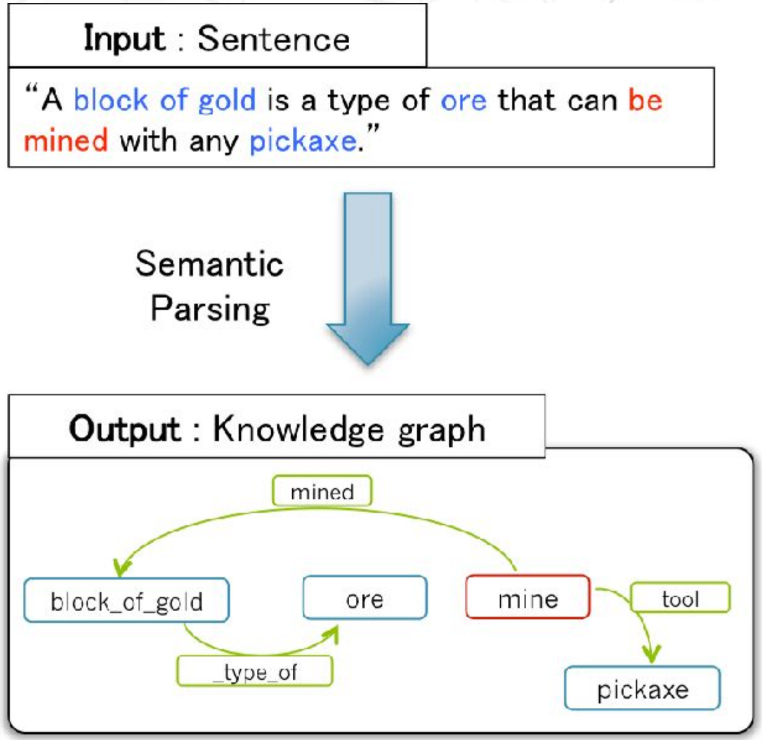
$\circ\varphi$ translates to $\langle true\rangle\varphi$;
$\bullet\varphi$ translates to $\neg\langle true\rangle\neg\varphi = [true]\varphi$;
$\Diamond\varphi$ translates to $\langle true^*\rangle\varphi$;
$\Box\varphi$ translates to $[true^*]\varphi$;
$\varphi_1 \mathcal{U} \varphi_2$ translates to $\langle(\varphi_1?; true)^*\rangle\varphi_2$.

Approaches:

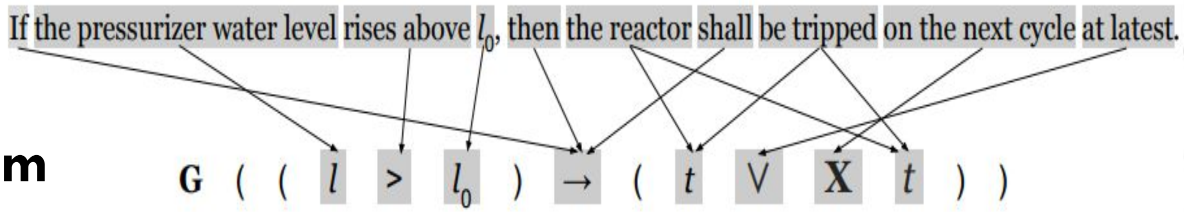
- **Semantic parsing**

  Common pattern of syntactic structure for controlled English

- Translation problem
- Optimization problem
- Event identification and ordering



Approaches:



- Semantic parsing
- **Translation problem**

  Logic is treated as a language and train NN/CNN for statistical models

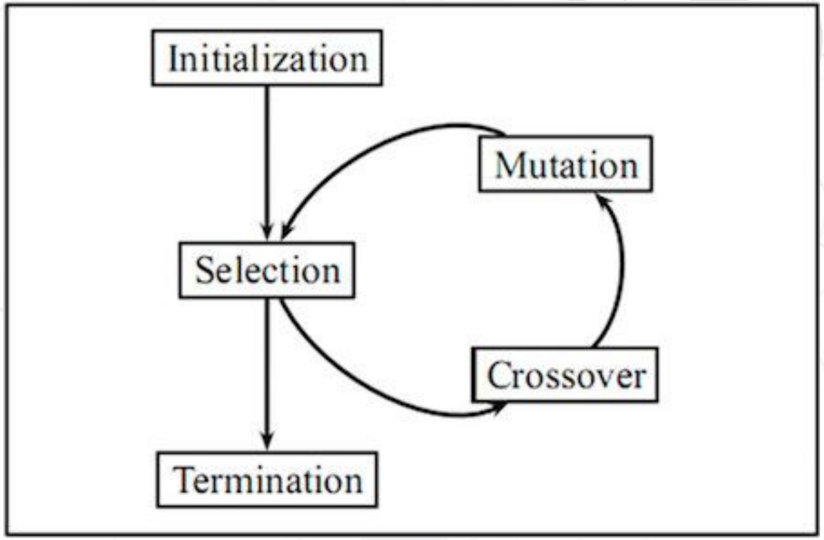
- Optimization problem
- Event identification and ordering

Approaches:

- Semantic parsing
- Translation problem
- **Optimization problem**

  Evolutionary Computation for determining best matching temporal formula

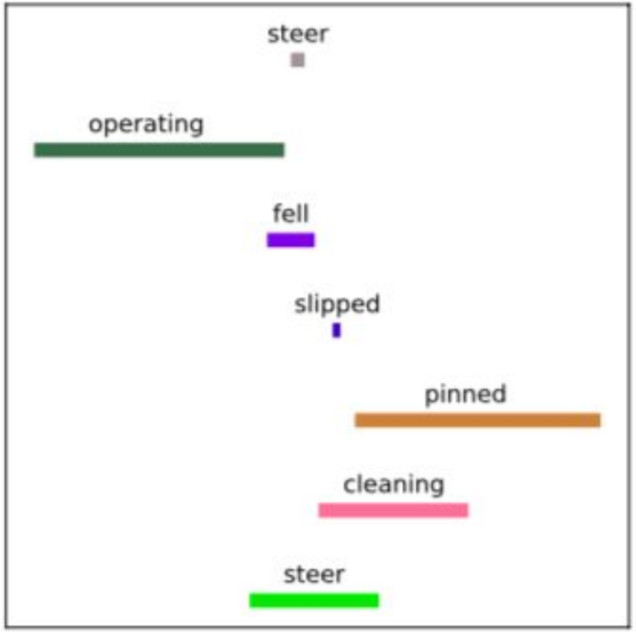
- Event identification and ordering



Approaches:

- Semantic parsing
- Translation problem
- Optimization problem
- **Event identification and ordering**

  Identify temporal characteristics and encode temporal order between events with graph



Approaches:

- **Semantic parsing**
- Translation problem
- Optimization problem
- Event identification and ordering

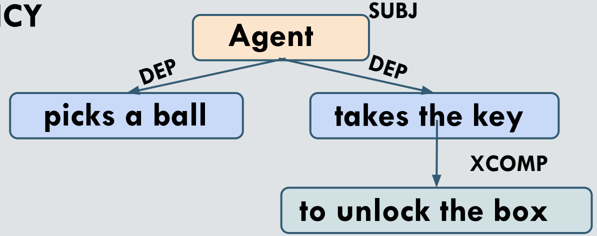
General scheme: Syntactic analysis

POS TAGGING:

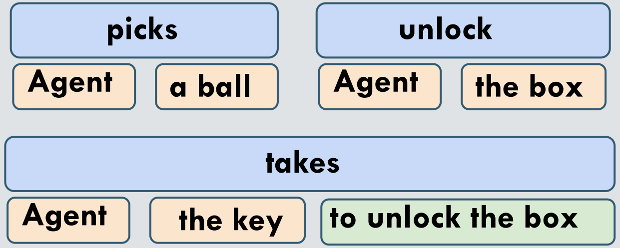
NN VB DT NN CJ VB DT NN PP VB DT NN

Agent picks a ball and takes the key to unlock the box

DEPENDENCY PARSING



SEMANTIC ROLE LABELING



Agent	(N1, o)	N1->Noun
picks	(V1, o)	V1->Tv
a ball	(N2,o)	N2->Det N2
picks a ball	(Rc1,o)	Rc1->N1 V3 N4
takes	(V2, o)	V2->Tv
the key	(N3,o)	N3->Det N3
unlock	(V3, o)	V3->Tv
the box	(N4,o)	N4->Det N4
unlock the box	(Rc2,o)	Rc2->N1 V3 N4
takes the key to unlock the box	(Rc2,o)	Rc2->N1 V2 N3 Rc2
final sentence	(Rc3,o)	Rc3->Rc1 Rc2

General scheme: grammar-based parser

Agent	(N1, o)	N1->Noun
picks	(V1, o)	V1->Tv
a ball	(N2,o)	N2->Det N2
picks a ball	(Rc1,o)	Rc1->N1 V3 N4
takes	(V2, o)	V2->Tv
the key	(N3,o)	N3->Det N3
unlock	(V3, o)	V3->Tv
the box	(N4,o)	N4->Det N4
unlock the box	(Rc2,o)	Rc2->N1 V3 N4
takes the key to unlock the box	(Rc2,o)	Rc2->N1 V2 N3 Rc2
final sentence	(Rc3,o)	Rc3->Rc1 Rc2

$$
\begin{aligned}
tr(a) &= true; a? \\
tr(\phi?) &= \phi? \qquad (\phi \text{ propositional}) \\
tr(\delta_1;\delta_2) &= tr(\delta_1); tr(\delta_2) \\
tr(\textbf{if}\ \phi\ \textbf{then}\ \delta_1\ \textbf{else}\ \delta_2) &= \phi?; tr(\delta_1) + \neg\phi?; tr(\delta_2) \\
tr(\textbf{while}\ \phi\ \textbf{do}\ \delta) &= (\phi?; tr(\delta_1))^*; \neg\phi?
\end{aligned}
$$

<
(!ball&!key&!box)*;ball;
(!ball&!key&!box)*;key;
(!ball&!key&!box)*;box;
> tt

3. Imitation Learning

Understanding the problem

Generate DFA for the expert agent

- Temporal logic definition of agent Task
- Reactive Synthesis
- LDLf to DFA translation

Use Restraining Bolt to train the expert agent on Q

- Restraining Bolt
- Fluent traces generation

Train LEarner Agent on Expert Policy

- Imitation Learning (Inverse Reinforcement Learning)
- Learner DFA Exraction

Generate DFA from LDLf formulas

Task definition

In synthesis typically agent tasks are expressed in terms of traces in temporal logic (LDLf in our case)

A trace τ is a finite (LDLf) sequence of fluents ∪ actions evaluations.

Example:

- Task: "unlock the door and get the box"
- <(¬ key ∧ ¬ door ∧ ¬ box)* ; key ; <(¬ key ∧ ¬ door ∧ ¬ box)* ; door ; <(¬ key ∧ ¬ door ∧ ¬ box)* ; box >tt
  - φ* (*safety*) : means that always, until the end of trace, φ holds.

    In example, φ = (¬ key ∧ ¬ door ∧ ¬ box)

  - true*; φ_1 ; true* ; φ_2 ; true* ; φ_3 (*ordered occurrence*): says that φ_1 and φ_2 and φ_3 will happen in order.

    In example: φ_1 = key | φ_2 = door | φ_3 = box

  - ⟨ρ⟩φ : all "executions" of RE ρ (along the race) end with φ holding.

    In example: φ = tt (true) | ⟨ρ⟩ = <(¬ key ∧ ¬ door ∧ ¬ box)* ; key ; <(¬ key ∧ ¬ door ∧ ¬ box)* ; door ; <(¬ key ∧ ¬ door ∧ ¬ box)* ; box >

Generate DFA from LDLf formulas

Synthesis

Given an LDLf task '*Task*' for the agent:

- Find agent behavior σ_a such that ∀σ_e, trace (σ_a , σ_e)|= *Task*

Agent Behavior

Also called "strategy", "policy", "protocol", "process":
σ_a : (*fluents*)* → *actions*

where:

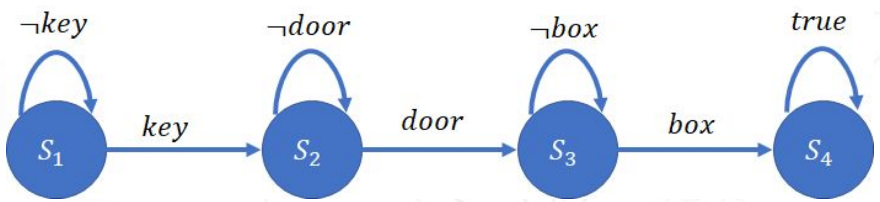
- (*fluents*)* denotes the **history** of what observed so far by the
- agent *(a finite sequence of fluents configurations)*
  actions denotes the **next action** that the agent does

Generate DFA from LDLf formulas

LDLf to DFA translation

- LTLf /LDLf formulas can be translated into deterministic finite state automata (DFA)

- t |= φ iff t ∈ $\mathcal{L}(A\_\varphi)$, where:
  - t is the trace of fluents;
  - A_φ is DFA φ is translated into ;
  - φ is the LDLf formula



DFA generated from LDLf formulas

Using RB to train expert agent on DFA

Restraining Bolt

A Restraining Bolt (RB) is a tuple $\mathrm{RB} = \langle \mathcal{L}, \{(\varphi_i, r_i)\} \rangle$ where each phi_i is an LDLf $\backslash$ LTLf formula over a set of fluents L and each r_i is a reward value.

Fluents constitute the RB's representation of the environment state and need not match the RL agent features (and typically they do not). Formulas phi_i specify the behaviors that should be rewarded, each with its respective r_i. As known (De Giacomo and Vardi 2013) LDLf / LTLf formulas can be equivalently represented as DFAs, and this representation is used to constrain an agent's behavior to fulfill high-level (i.e., fluent-based) goals.

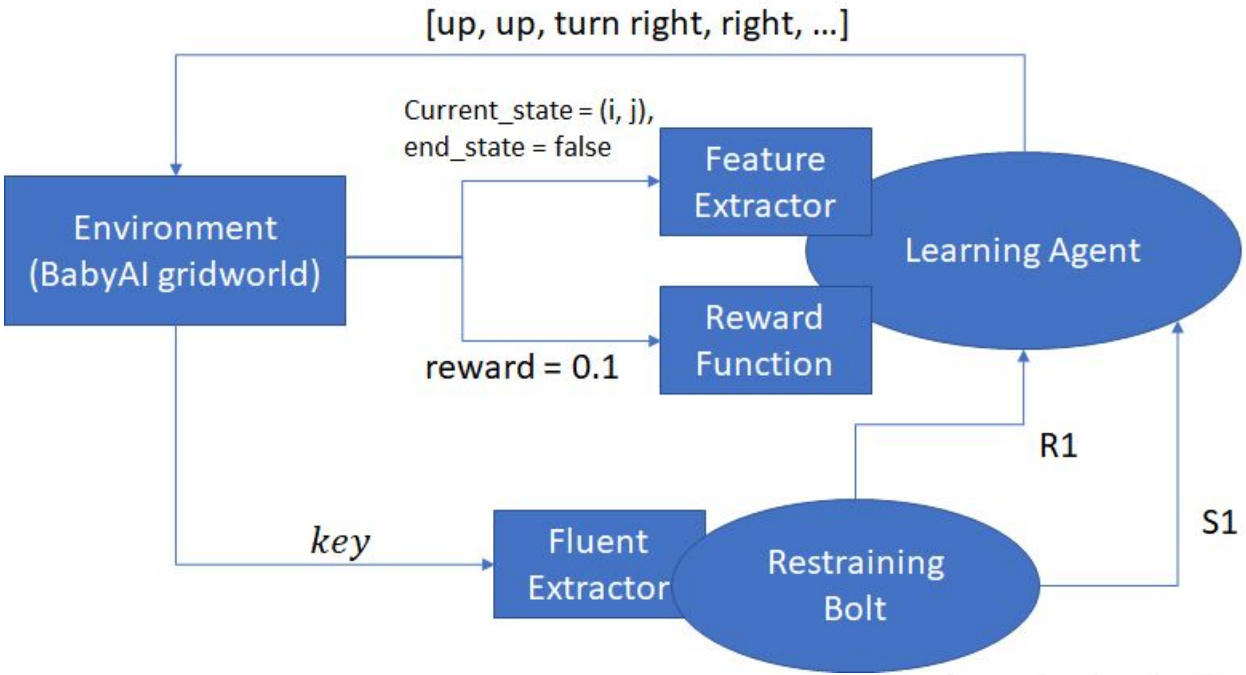
Using RB to train expert agent on DFA

Thus the task is represented by the DFA Q corresponding to a formula phi_i. As a result, we consider RBs of the form <L, Q, r_i> where Q is a DFA representing an LTLf/LDLf formula and r is a reward value associated with the accepting states of Q.

Consider now an expert agent defined on an MDP Me = <Se; Ae; T re; Re >. The agent can execute optimal policies of a given target task represented by a DFA Q, but cannot make the corresponding reward function explicit; in other words, the agent knows how to accomplish the task but cannot describe it.

As the agent executes the policy, some traces are produced, some of which are desirable (posiive) and some other are not. The expert can correctly classify the traces as positive or negative, based on its own state representation.

Using RB to train expert agent on DFA



Train learner agent on expert policy

Imitation Learning

In our project we focused on Inverse Reinforcement Learning (IRL) approach for the implementation of the IL algorithm, as discussed in the paper "*Imitation Learning over Heterogeneous Agents with Restraining Bolts*" (De GIacomo, Favorito, Iocchi, Patrizi) .

The main idea of IRL is to learn the reward function of the environment based on the expert's demonstrations, and then find the optimal policy (the one that maximizes this reward function) using reinforcement learning

Thus the RB device is attached to the learner agent to drive the learning process and ultimately make it imitate the expert.

Train learner agent on expert policy

Learner DFA Extraction

Let $\mathcal{T}$ be a set of fluent traces collected while observing the behavior of the expert. The imitation learning process consists in recostructing a DFA $Q_{\mathcal{T}}$ that is consistent with $\mathcal{T}$, i.e., that accepts all of its positive traces and none of its negative. Thus is a new RB:

$$\mathrm{RB} = \langle \mathcal{L}, Q_{\mathcal{T}}, r \rangle$$

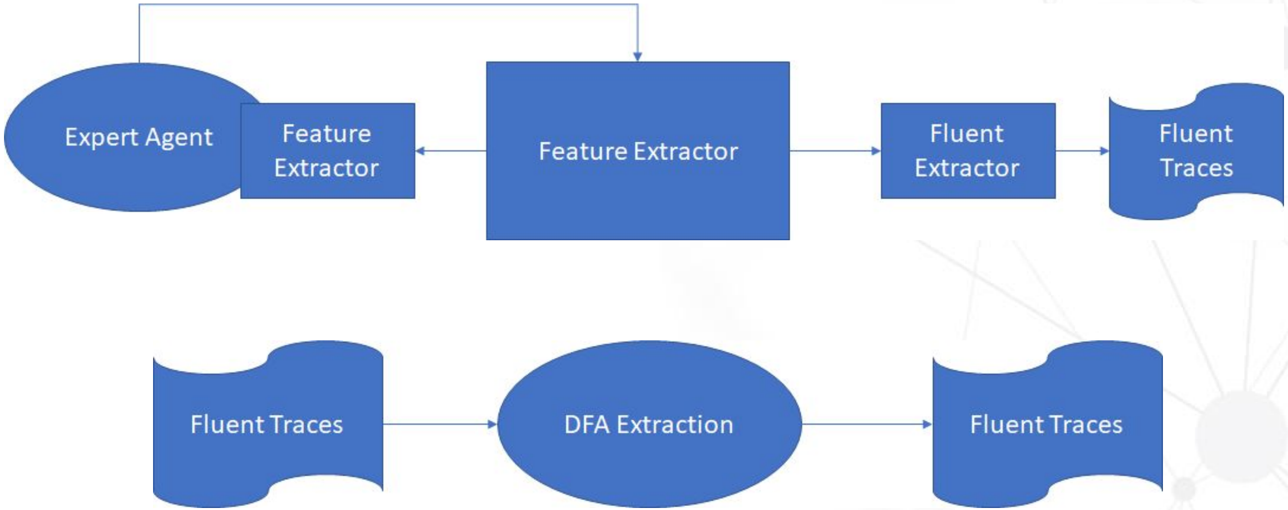
RB's DFA learning setting

The generated RB can be placed on the learner agent to drive the learning process of a behaviour imitating the expert's.

Train learner agent on expert policy

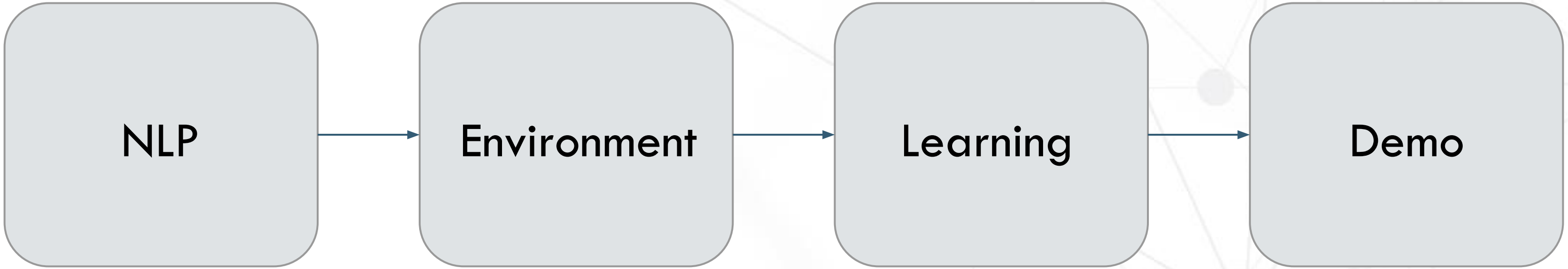
Consider a learner agent defined on MI = <S_I , A_I , Tr_I , R_I >, with Tr_I and R_I unknown, equipped with the RB that encodes the behavior of the expert agent in performing the given task. The system M_I^RB = < MI , RB > can be used to learn an optimal policy driven by RB, as explained in (De Giacomo et al. 2019). In this way, the behavior of the learner agent imitates that of the expert, when considering the evolution at the RB level.

Train learner agent on expert policy



4. Implementation

Summary



Natural Language Processing

SPEAKING

Acquire sentence by listening to the speaker

TRANSCRIBING

Transcribe what the speaker has said

POS TAGGING

Apply Part Of Speech tagging to the sentence in order to be able to create links between verbs and objects

TRANSLATING

Translate the NL sentence into LDLf

NLP to LDLf

```
The Goal:
Agent picks a ball and takes the key to unlock the box
['Agent', 'pick', 'a', 'ball', 'and', 'take', 'the', 'key', 'to', 'unlock', 'the', 'box']
['NN', 'VB', 'DT', 'NN', 'CC', 'VB', 'DT', 'NN', 'TO', 'VB', 'DT', 'NN']
Reduced version:
['Agent', 'pick', 'ball', 'and', 'take', 'key', 'unlock', 'box']
['NN', 'VB', 'NN', 'CC', 'VB', 'NN', 'VB', 'NN']
LDLf formuls:
<(!ball & !key & !box)*;ball;(!ball & !key & !box)*;key;(!ball & !key & !box)*;box>tt

In [12]:
```

Environment

In order to replicate as much as possible the environment of BabyAI, we focused on the actions that our agent would implement.

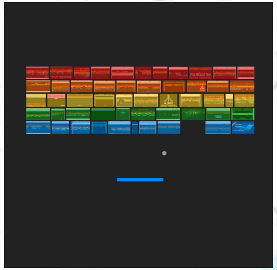
Requests: Get the box

Open the door

Move the ball

Use the key

...



Actions and objects

In its basic version, Minecraft defines two main actions: get and use. A third action to manage **pick and drop** situations is added to these two.

GET

USE

MOVE

All the objects provided by Minecraft, were replaced by typical objects of BabyAI. Another category of objects is also added.

Resources:	DOOR	
	BOX	
	BALL	
	BASE	
Tools:	KEY	
Obstacles:	WALL	

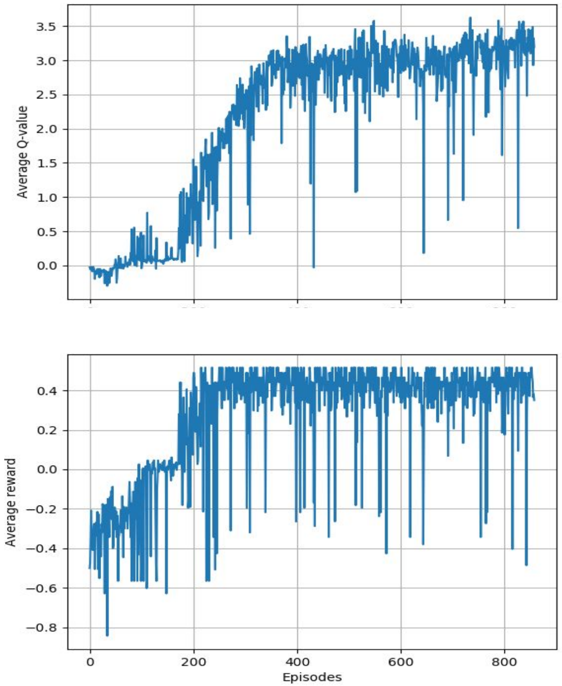
Learning

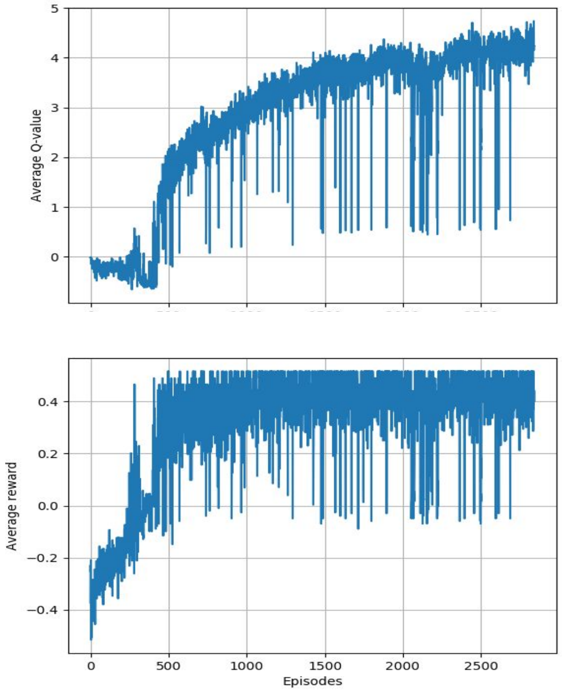
Use the key, open the door and get the box.

Positive Traces	Negative Traces
key; door; box	key; key
key; door; box	door
key; door; box	key; box
key; door; box	key; key
…	…

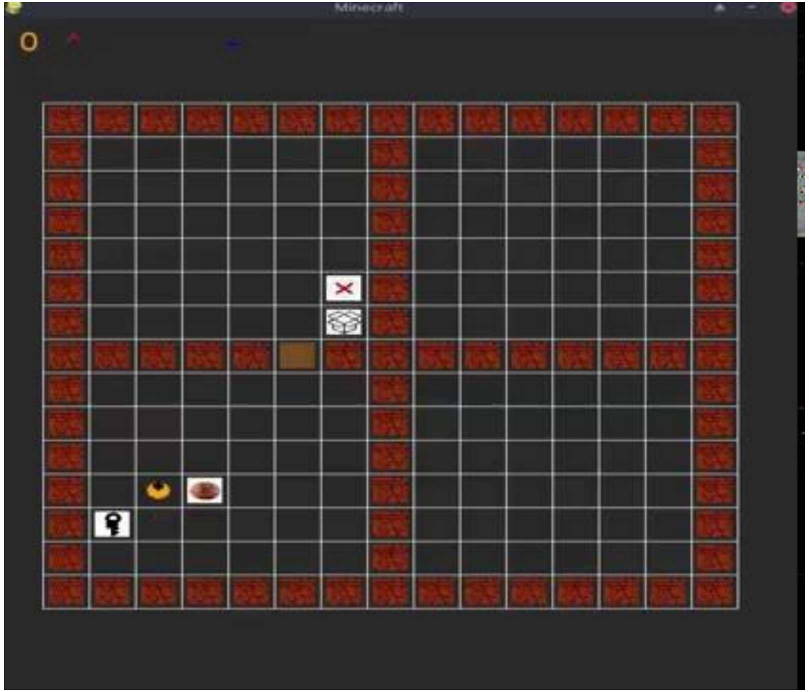
Expert	**Params**	**Learner**
30000	num_steps	100000
0.1	min_eps	0.1
0.99	gamma	0.99
0.1	alpha	0.1
sarsa	algorithm	sarsa

Learning


Average Q-value
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Average reward
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
200
400
600
800
Episodes


Average Q-value
5
4
3
2
1
0
Average reward
0.4
0.2
0.0
−0.2
−0.4
0
500
1000
1500
2000
2500
Episodes

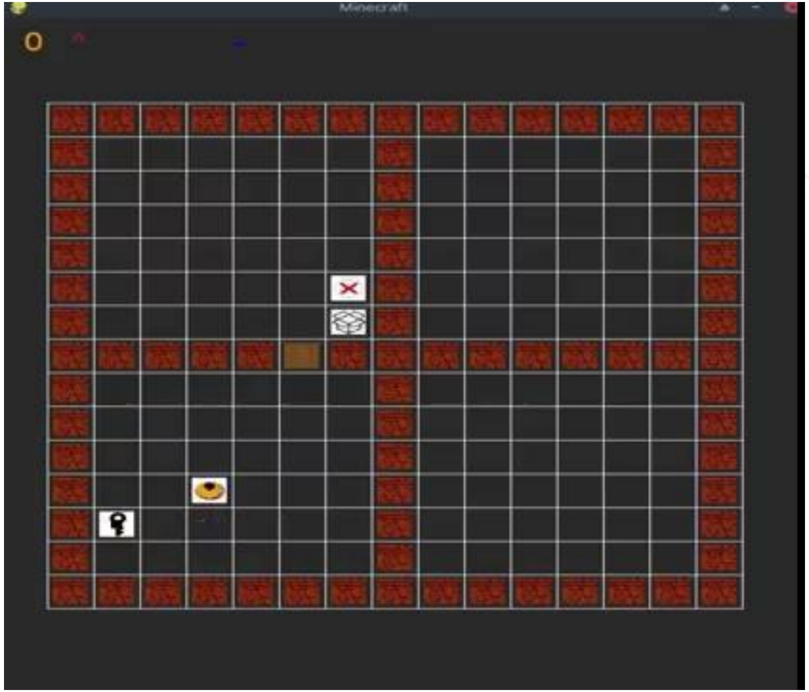
Demo - Level 1



Task: Use the key and get the box

The agent must unlock the door before accessing the second room where the box is placed. Once it uses the key, the door gets unlocked and it is no more an obstacle.

Demo - Level 2



Task: Use the key and move the ball

The agent must unlock the door before accessing the second room where the base is placed. Once it uses the key, the door gets unlocked and it is no more an obstacle. The ball is also moved from the initial position to the base.

Conclusions

BabyAI-like environment implementation ✔

LDLf and Restraining Bolt integration ✔

Combination of multiple tasks ✔

Future works: Usage of BabyAI to make the agent understand LDLf

References

- BABYAI: A Platform to study the sample efficiency of grounded language learning

  [Chevalier-Boisvert, Bahdanau, et al. - 2019]

- Gated-Attention Architectures for Task-Oriented Language Grounding

  [Chaplot, Sathyendra - 2018]

- Synthesis of LTL formulas from natural language texts: State of the art and research directions

  [Andrea Brunello, Angelo Montanari, Mark Reynolds - 2019]

- Imitation Learning over Heterogeneous Agents with Restraining Bolts

  [De Giacomo, Marco Favorito, Luca Iocchi, Fabio Patrizi - 2020]

Thank you!