Program Synthesis using Deduction-Guided Reinforcement Learning

Yanju Chen, Chenglong Wang, Osbert Bastani, Isil Dillig and Yu Feng

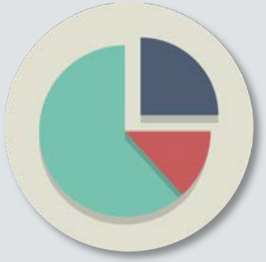
UCSB
UNIVERSITY of WASHINGTON
Penn UNIVERSITY of PENNSYLVANIA
TEXAS The University of Texas at Austin

Program Synthesis

specifications ϕ **synthesizer** **program P**

Find a program P that satisfies all the specifications ϕ.

table transformation



data visualization

string transformation

HPC

Program Synthesis

Deductive Reasoning

MORPHEUS
(Feng et al. 2017)

NEO
(Feng et al. 2018)

BLAZE
(Wang et al. 2018)

DISCONNECTION

DEEPCODER
(Balog et al. 2017)

AutoPandas
(Bavishi et al. 2019)

Statistical Reasoning

The feedback of deduction can not be seamlessly used by the statistical model.

There's a fundamental disconnection between program synthesis using pure statistical and deductive methods, so…

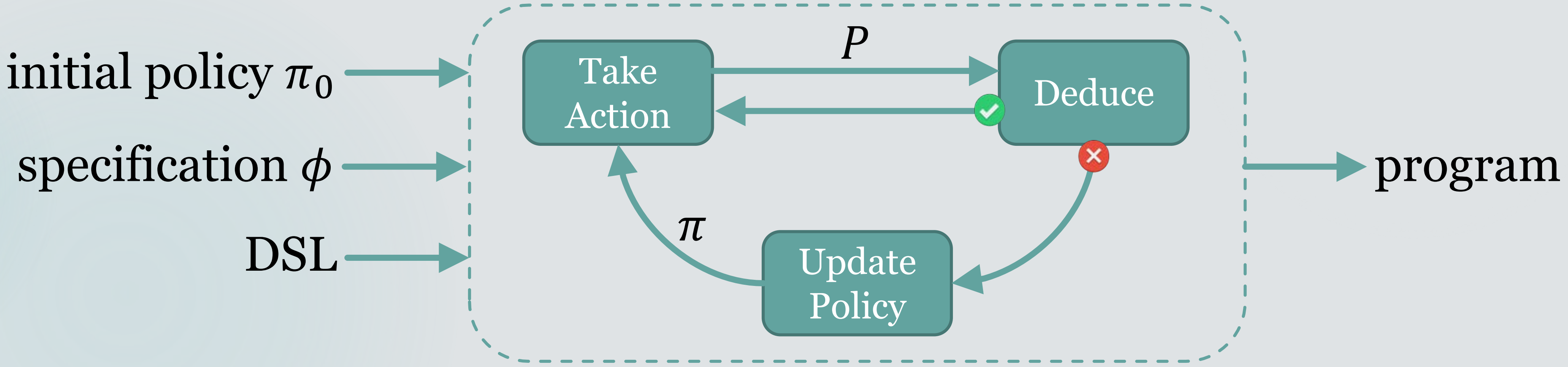
Can we bridge the statistical and deductive approaches in program synthesis in a seamless way?

CONCORD

utilizes fine-grained feedbacks from deductive reasoning
to update statistical reasoning on the fly

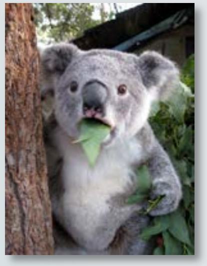
Our Approach: CONCORD

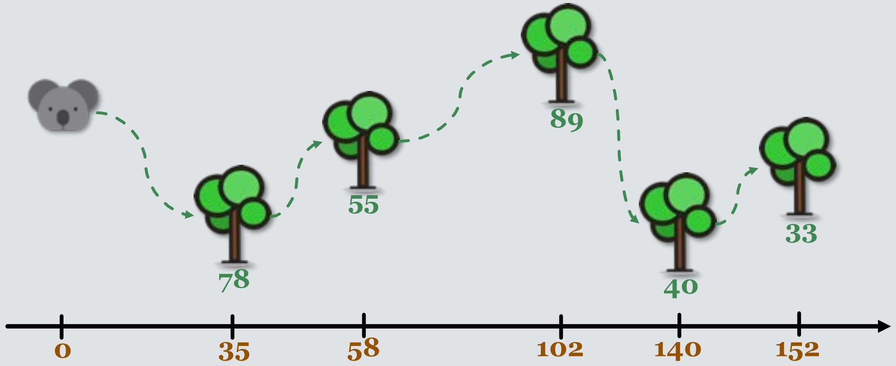
(An Overview)



A Running Example
(Koala Habitat Selectivity Study*)

- (goal) aggregate two koala factor tracking lists: `koalaFactor::list->list->list`
  - (input1) **factor 1** - GPS last seen locations (reversed order): `[152,140,102,58,35]`
  - (input2) **factor 2** - tree conditions: `[78,55,89,40,33]`





  - (output) model computes cumulative factors: `[43,75,120,122,143]`

[152 , 140 , 102 , 58 , 35]

[78 , 55 , 89 , 40 , 33]

*Synthetic data inspired by Davies, N., Gramotnev, G., Seabrook, L. *et al.* Movement patterns of an arboreal marsupial at the edge of its range: a case study of the koala. *Mov Ecol* **1**, 8 (2013).

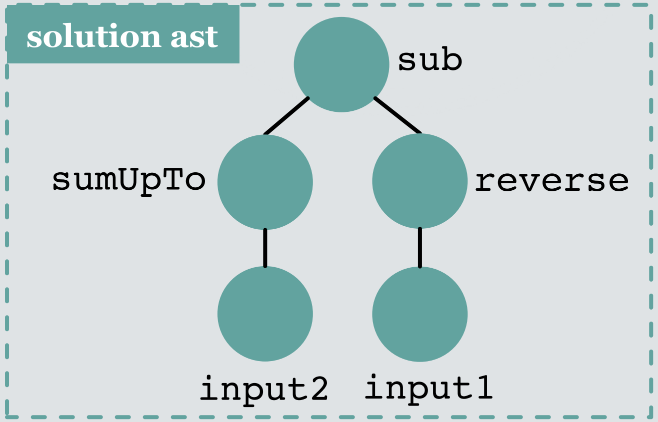
Sample Domain Specific Language

(A DSL for List Processing)

- **`koalaFactor::list->list->list`**
  - **`koalaFactor`**`([152,140,102,58,35],[78,55,89,40,33])=[43,75,120,122,143]`

sample DSL

```
S -> N|L
N -> 0|...|10|input1|inpu2
L -> input1|input2|take(L,N)|drop(L,N)|sort(L)
      |reverse(L)|add(L,L)|sub(L,L)|sumUpTo(L)
```

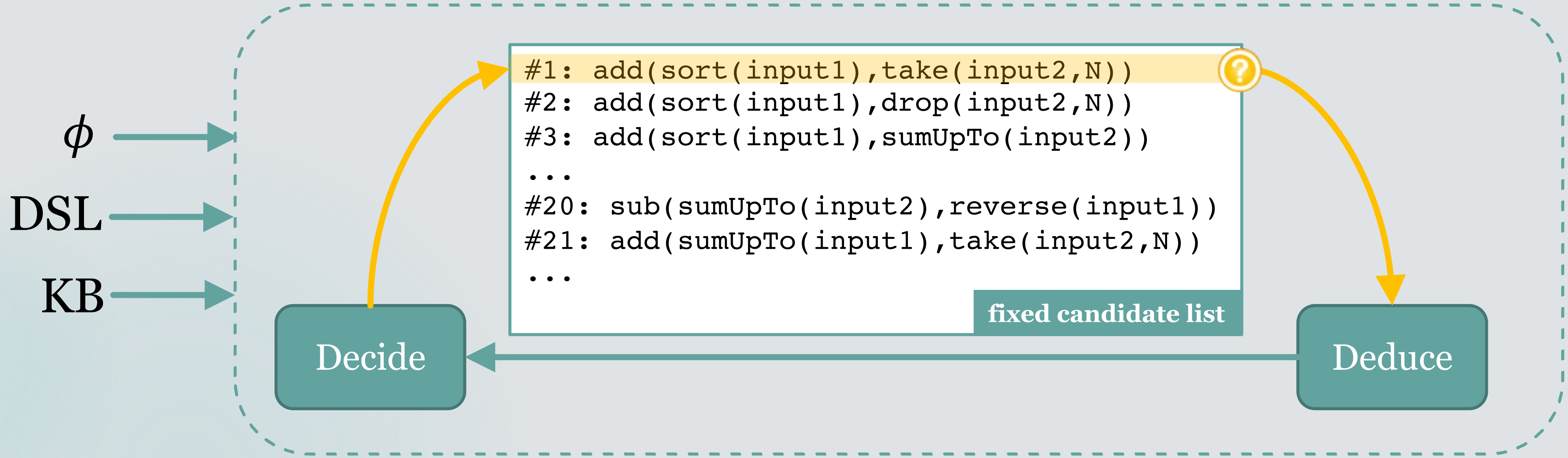


- solution program:
  - `v1 <-` **`reverse`** `input1`
  - `v2 <-` **`sumUpTo`** `input2`
  - `v3 <-` **`sub`** `v2,v1`
  - which is: **`sub(sumUpTo(`**`input2`**`), reverse(`**`input1`**`))`**

*CEGIS: Counter-Example-Guided Inductive Synthesis

Deductive Approach

(Solving `koalaFactor` in Variant of CEGIS* Loop, Step1)

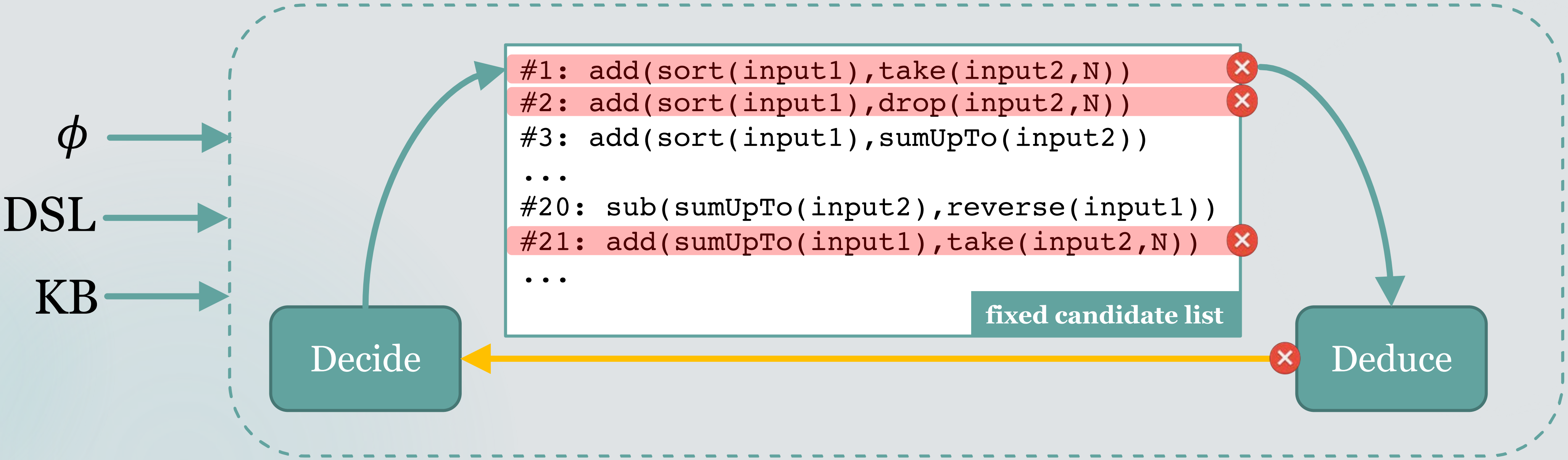



```
input1 = [152,140,102, 58, 35]
input2 = [ 78, 55, 89, 40, 33]
output = [ 43, 75,120,122,143]
```

*CEGIS: Counter-Example-Guided Inductive Synthesis

Deductive Approach

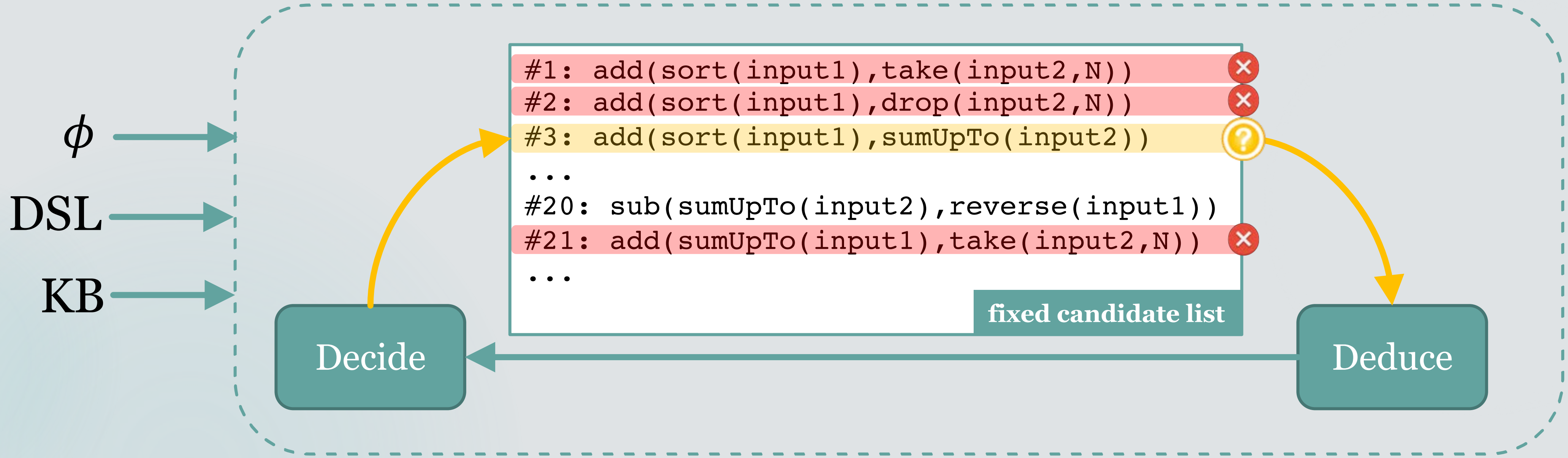
(Solving `koalaFactor` in Variant of CEGIS Loop, Step2)




```
input1 = [152,140,102, 58, 35]
input2 = [ 78, 55, 89, 40, 33]
output = [ 43, 75,120,122,143]
```

Deductive Approach

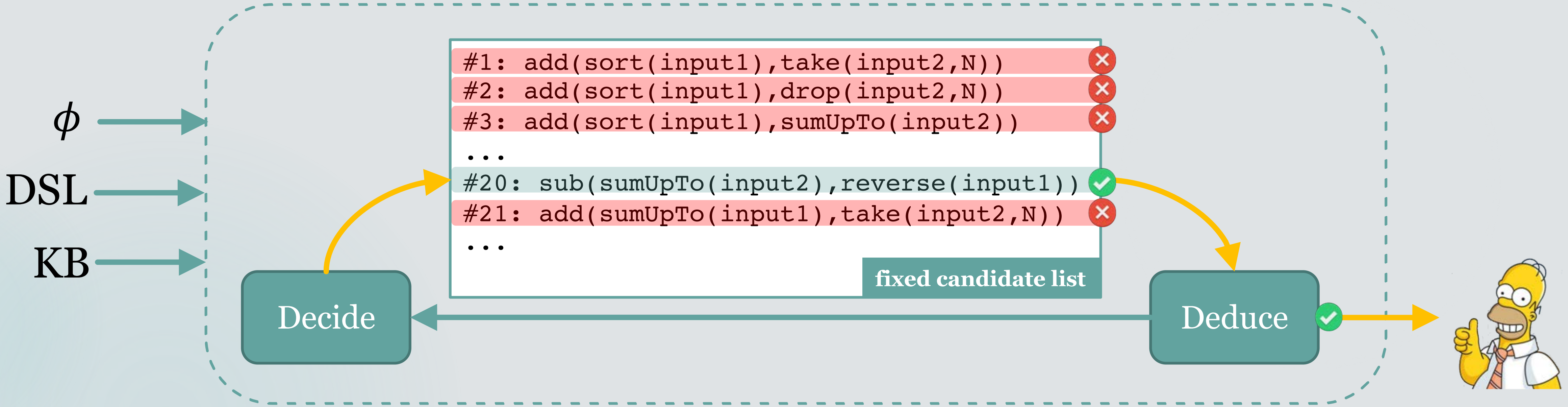
(Solving `koalaFactor` in Variant of CEGIS Loop, Step3)




```
input1 = [152,140,102, 58, 35]
input2 = [ 78, 55, 89, 40, 33]
output = [ 43, 75,120,122,143]
```

Deductive Approach

(Solving `koalaFactor` in Variant of CEGIS Loop, Step4)



- rich and accurate deduction feedback
- efficient pruning of search space

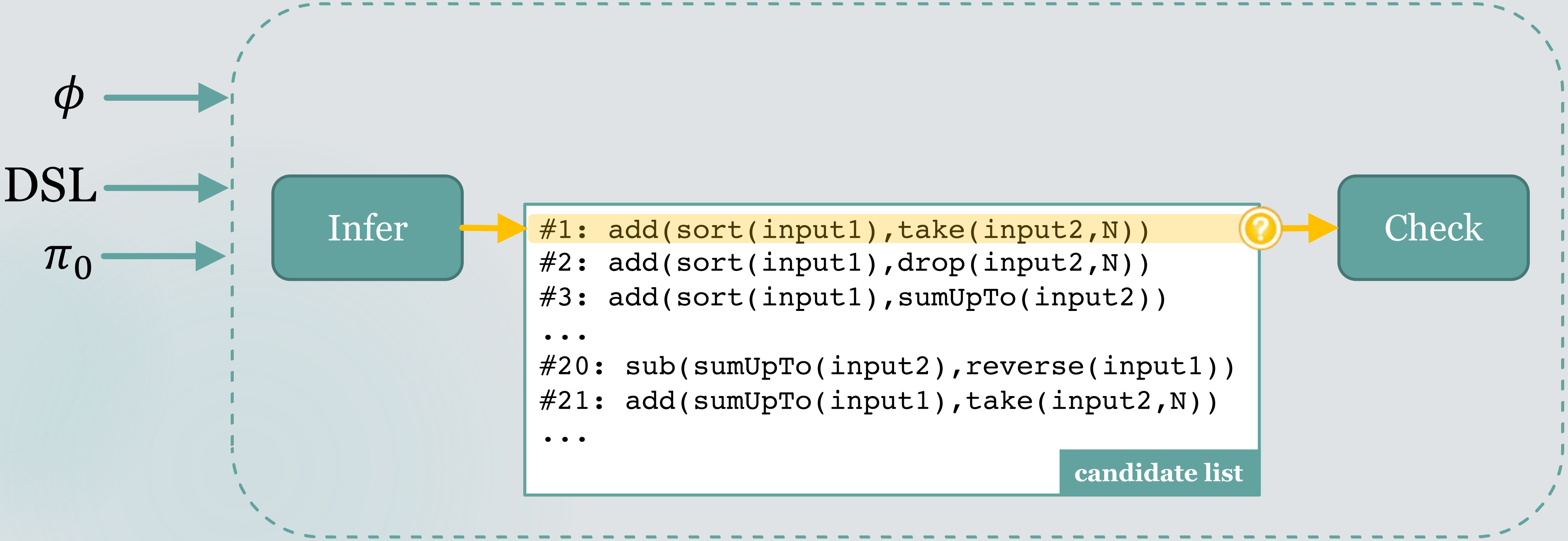
Pros

- incorporating deductive feedback into existing statistical model is difficult

Cons

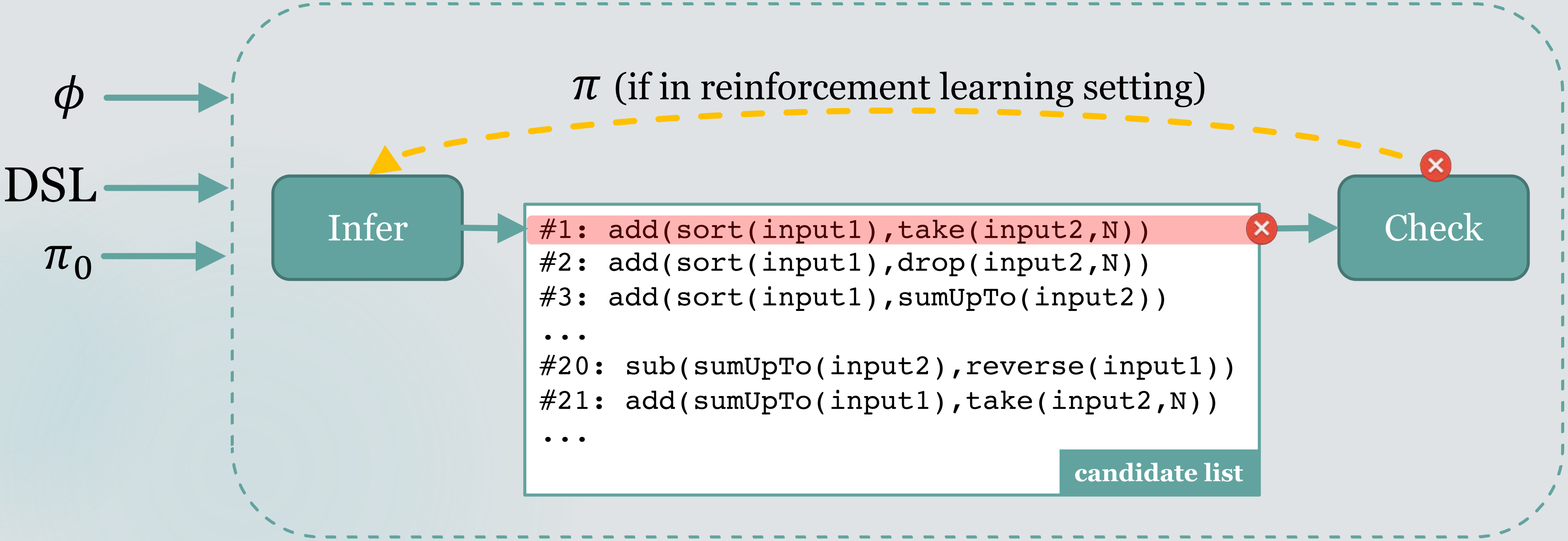
Statistical Approach

(Solving `koalaFactor` using Data-Driven Machine Learning, Step1)



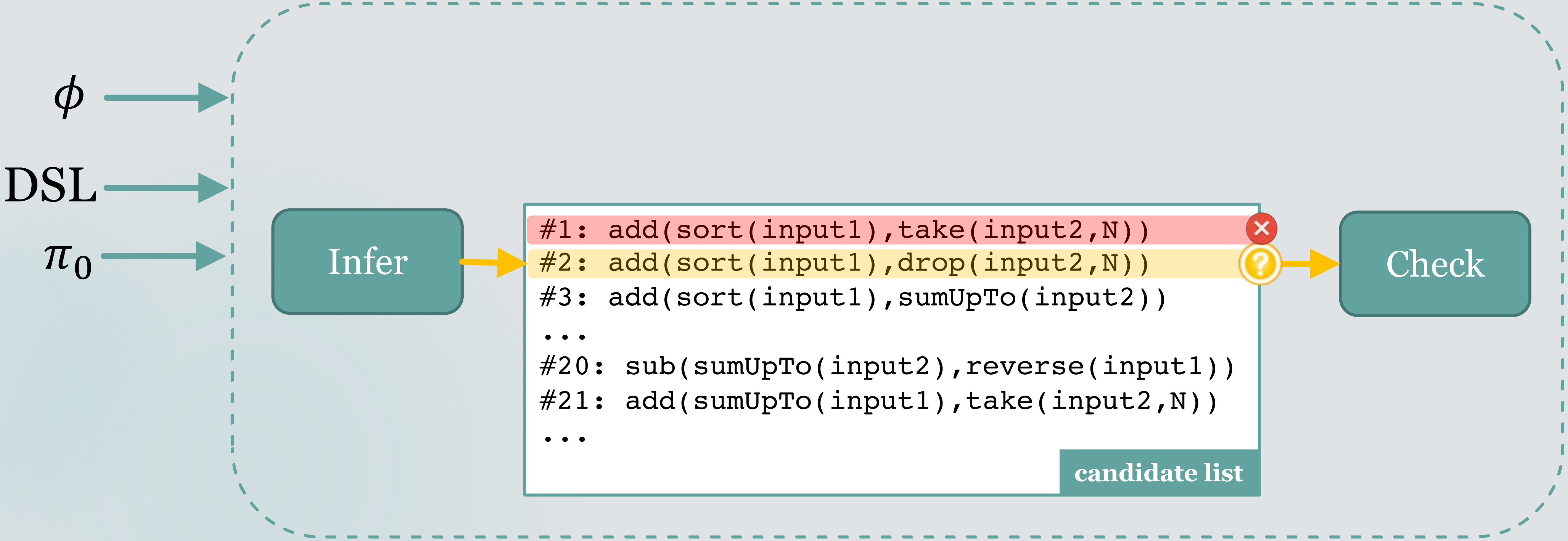
Statistical Approach

(Solving `koalaFactor` using Data-Driven Machine Learning, Step2)



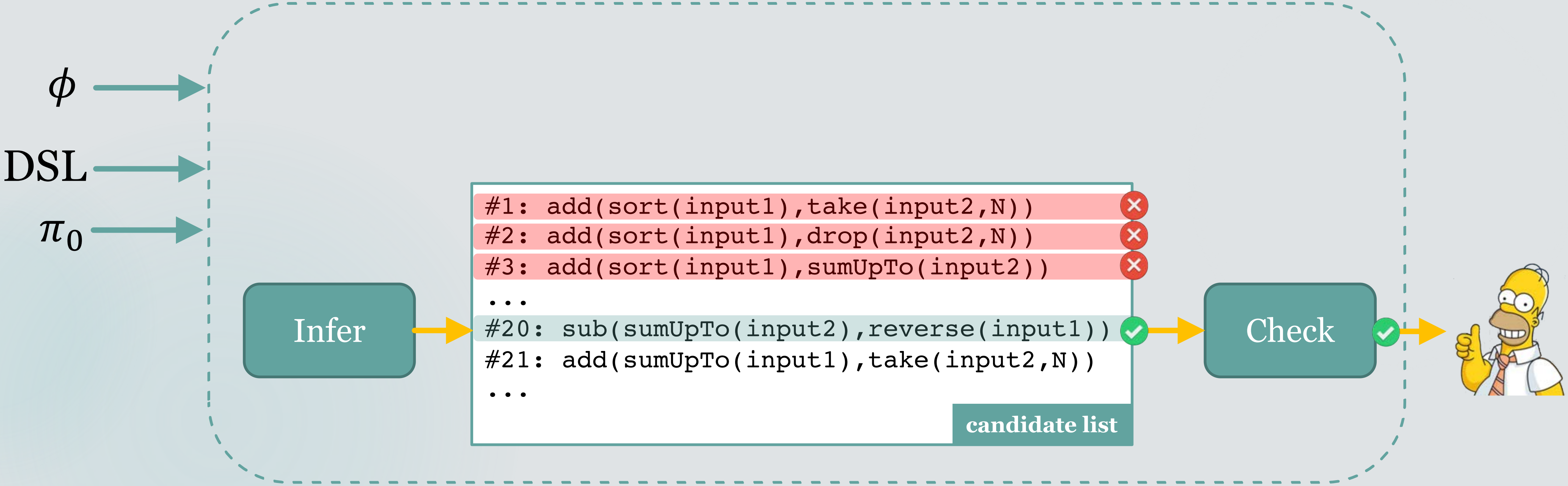
Statistical Approach

(Solving `koalaFactor` using Data-Driven Machine Learning, Step3)



Statistical Approach

(Solving `koalaFactor` using Data-Driven Machine Learning, Step4)



- data-driven candidate list
- can update policy seamlessly

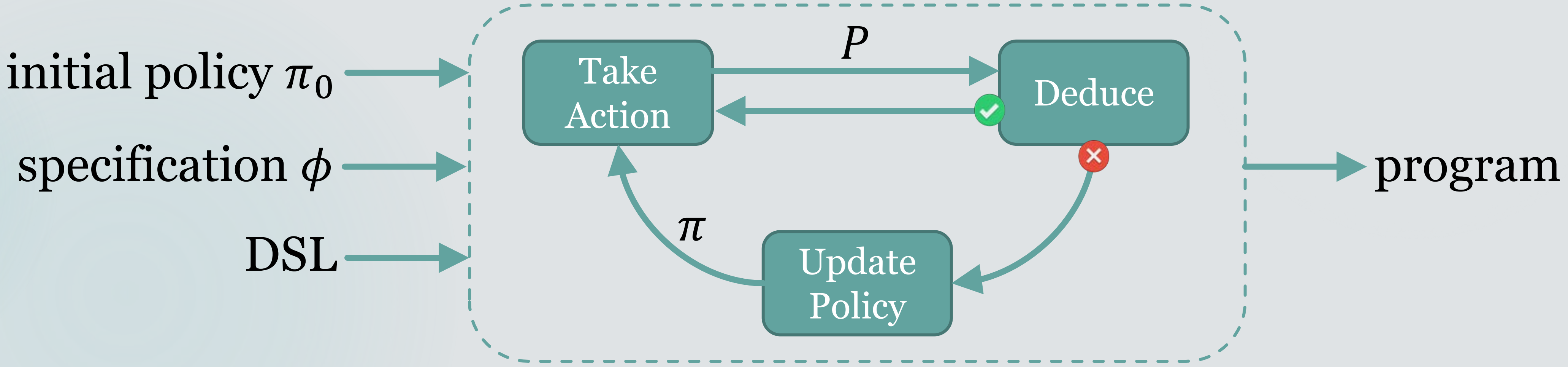
Pros

- feedback from checker is relatively less informative than that from deduction

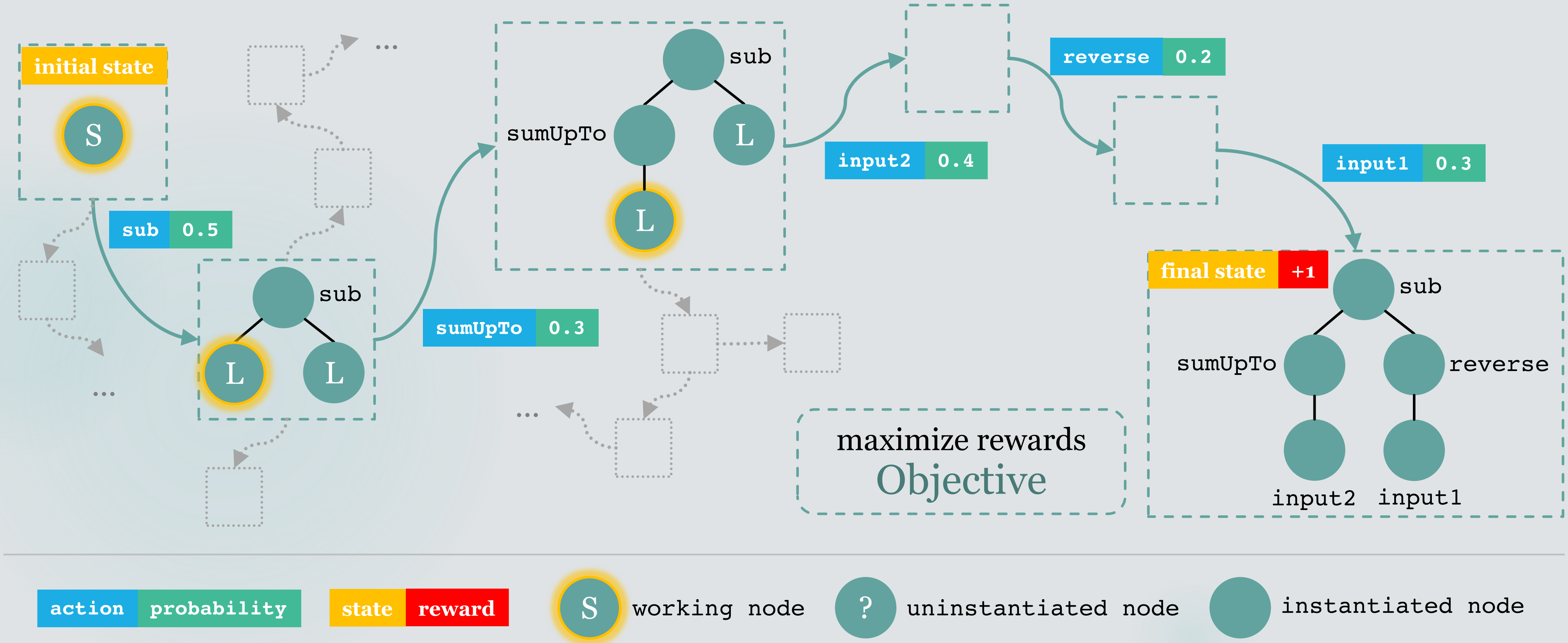
Cons

Our Approach: CONCORD

(An Overview)

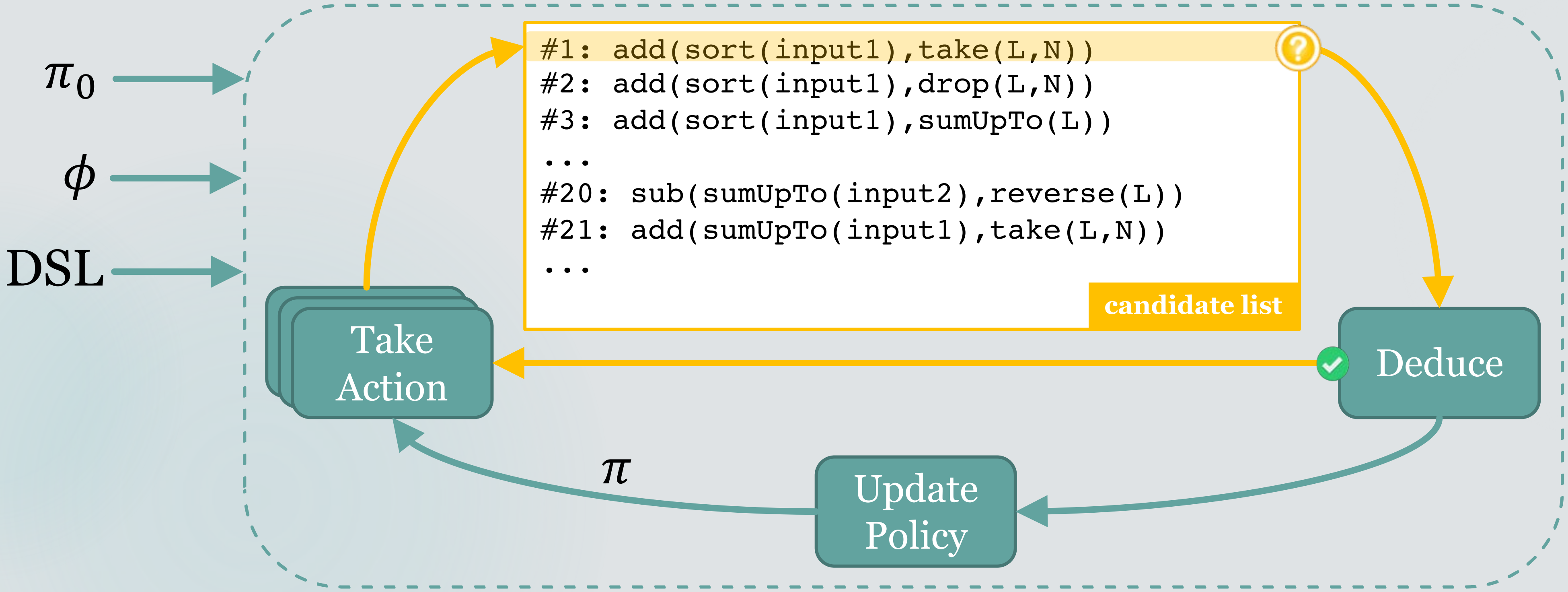


CONCORD: Formalization
(Program Synthesis as Markov Decision Process)



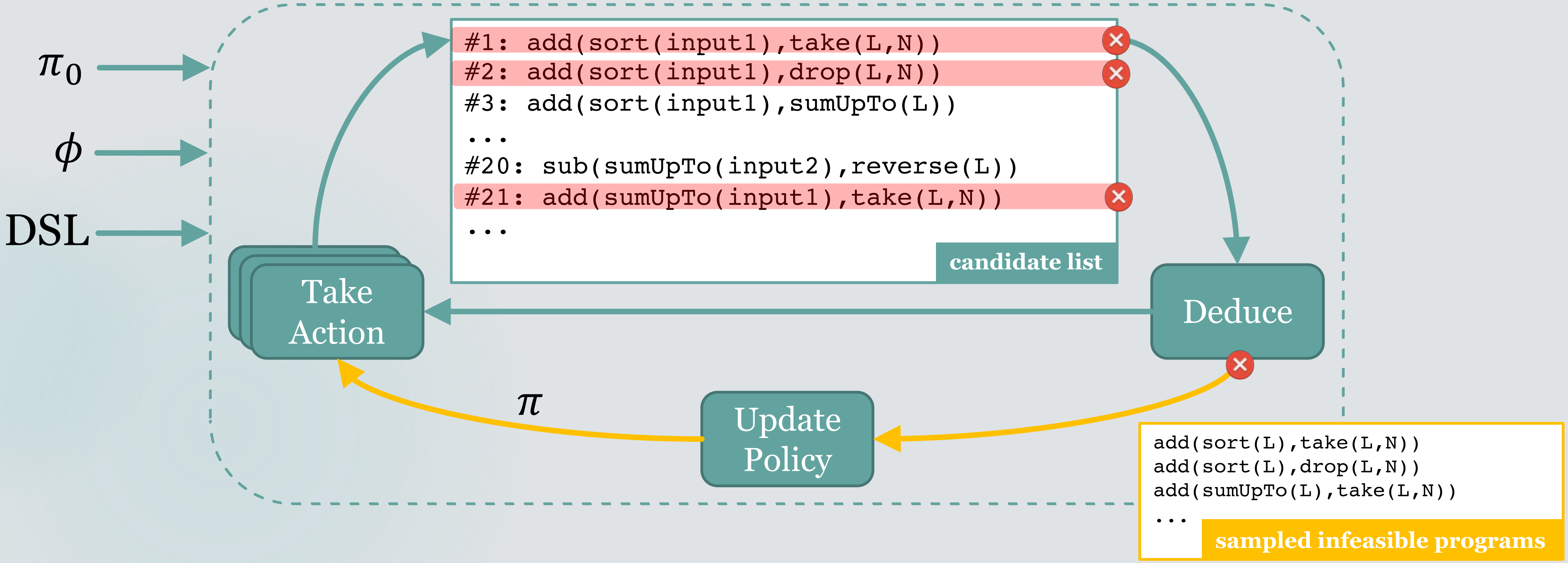
CONCORD: Running Example

(Solving `koalaFactor` using Deduction-Guided RL, Step1)



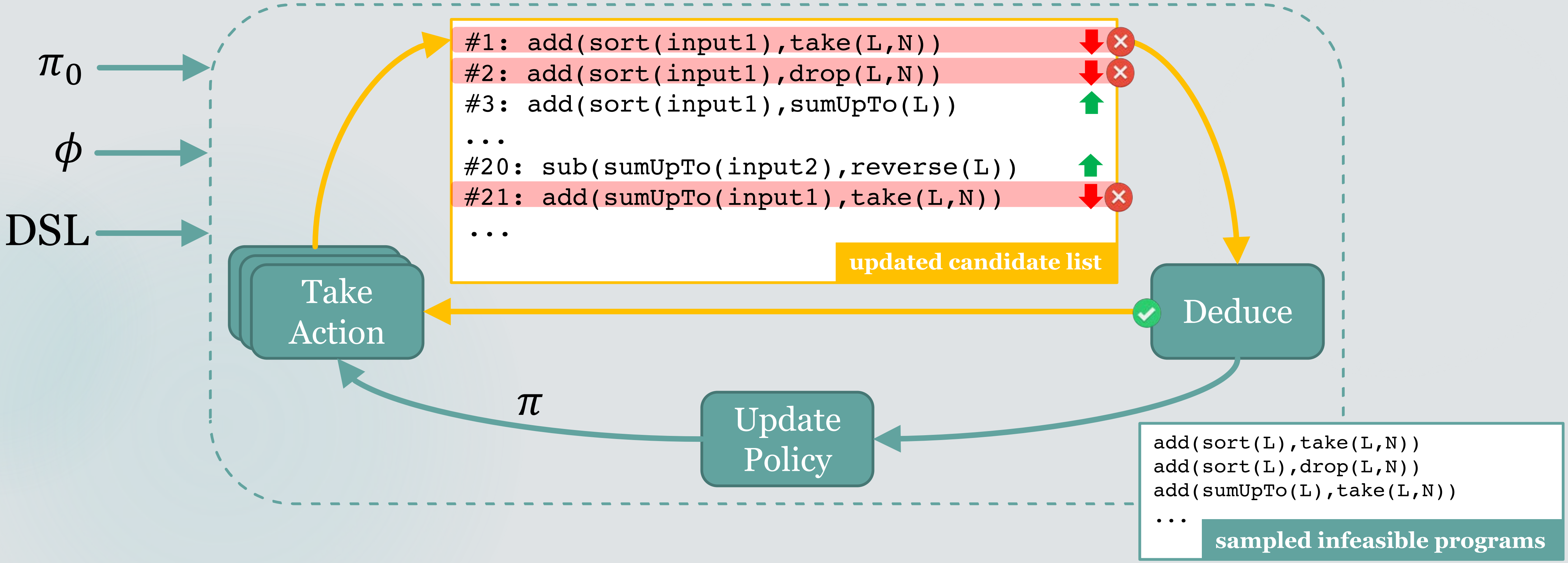
CONCORD: Running Example

(Solving `koalaFactor` using Deduction-Guided RL, Step2)



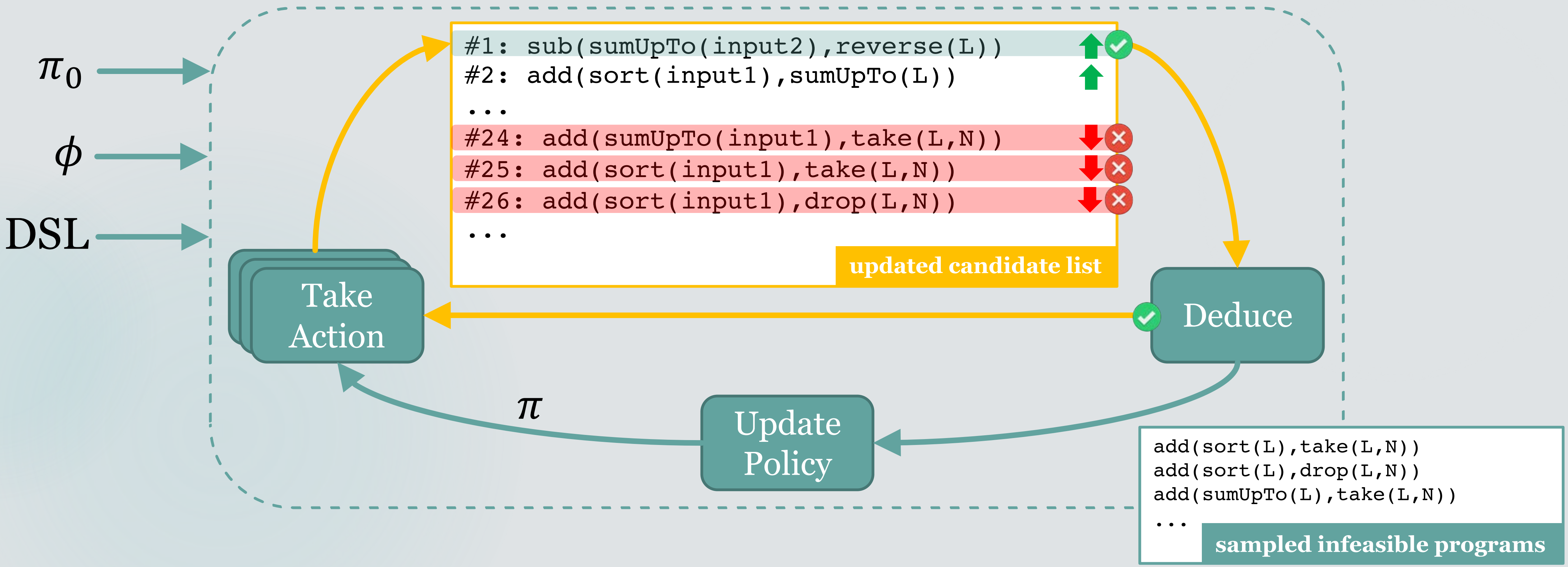
CONCORD: Running Example

(Solving `koalaFactor` using Deduction-Guided RL, Step3)



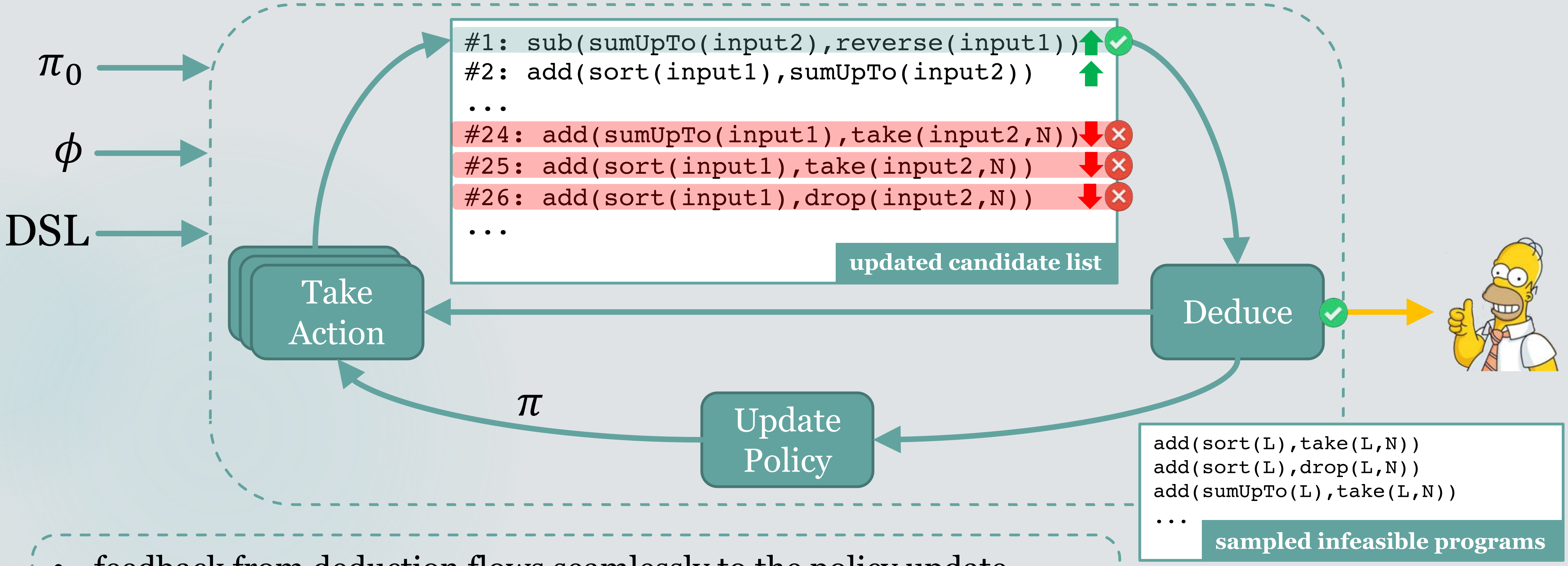
CONCORD: Running Example

(Solving `koalaFactor` using Deduction-Guided RL, Step3)



CONCORD: Running Example

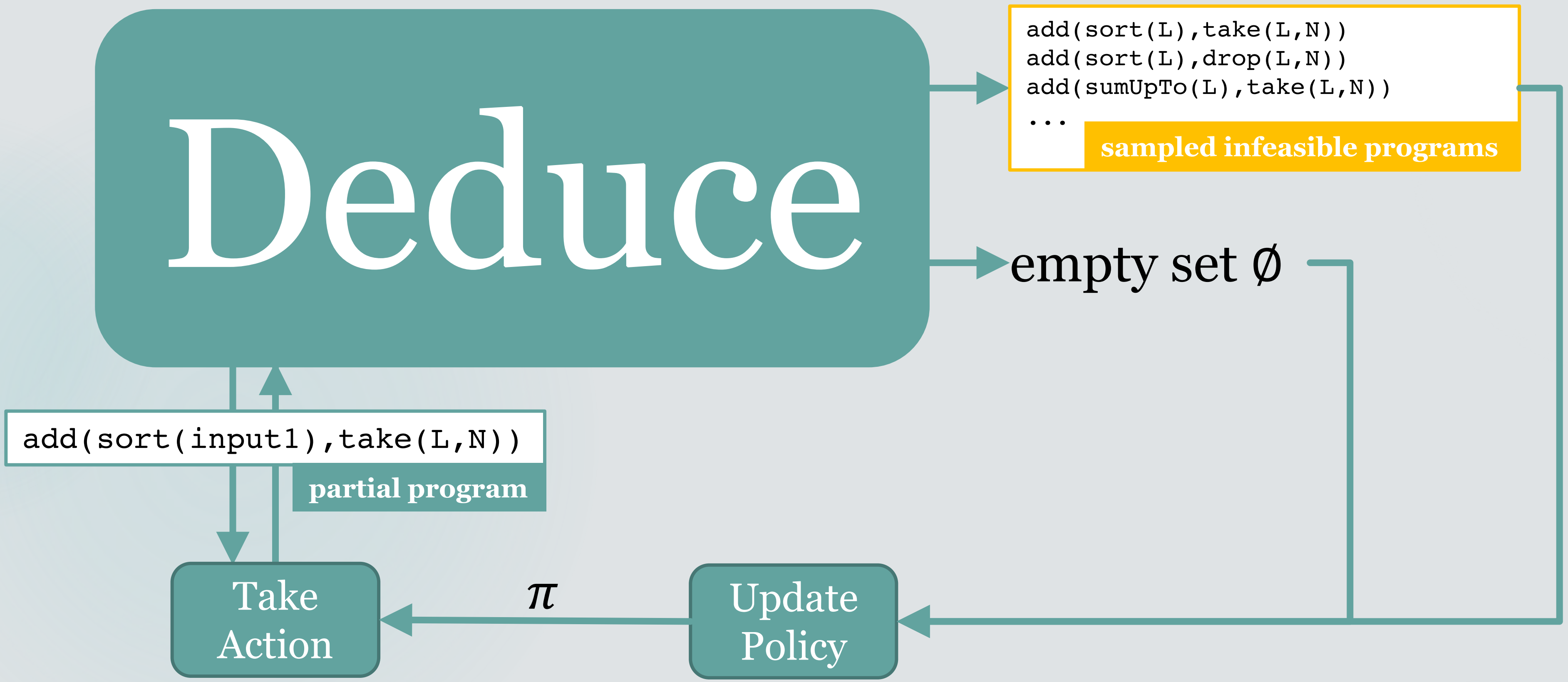
(Solving `koalaFactor` using Deduction-Guided RL, Step4)



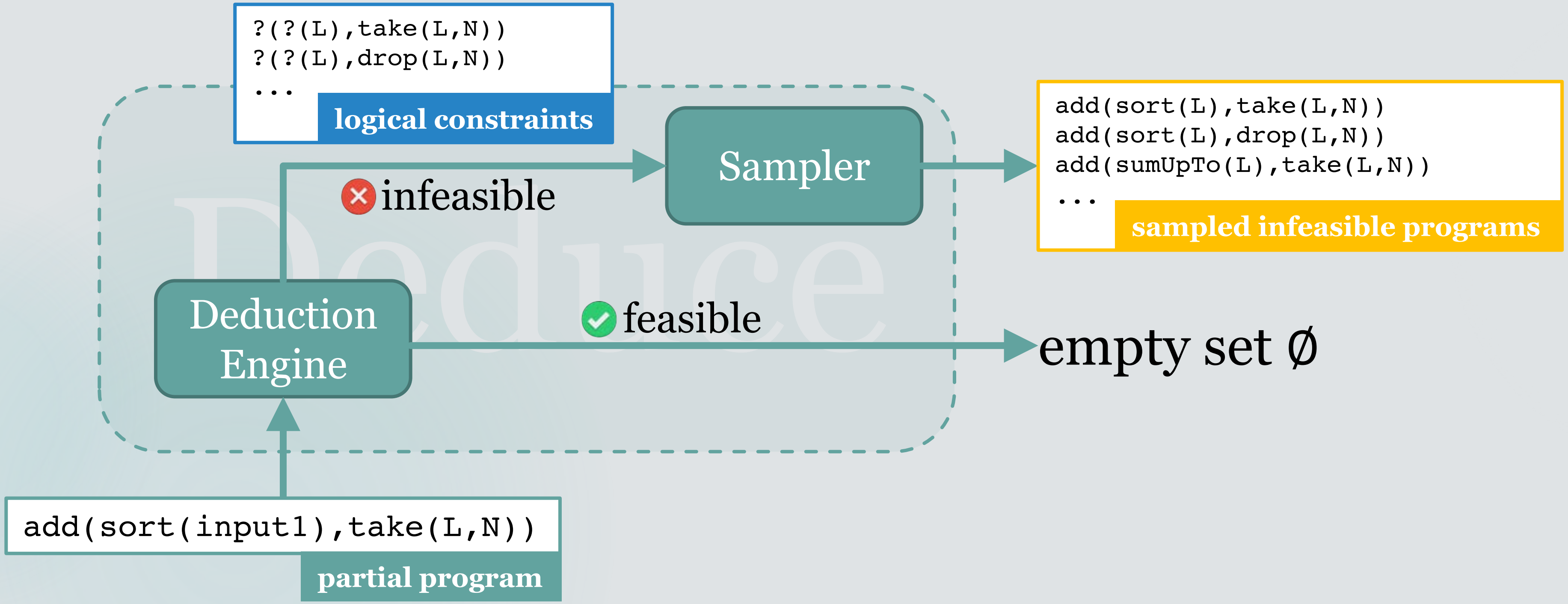
- feedback from deduction flows seamlessly to the policy update
- not only prune the search space, but also promote good candidates

What's the difference?

CONCORD: Synthesis Algorithm

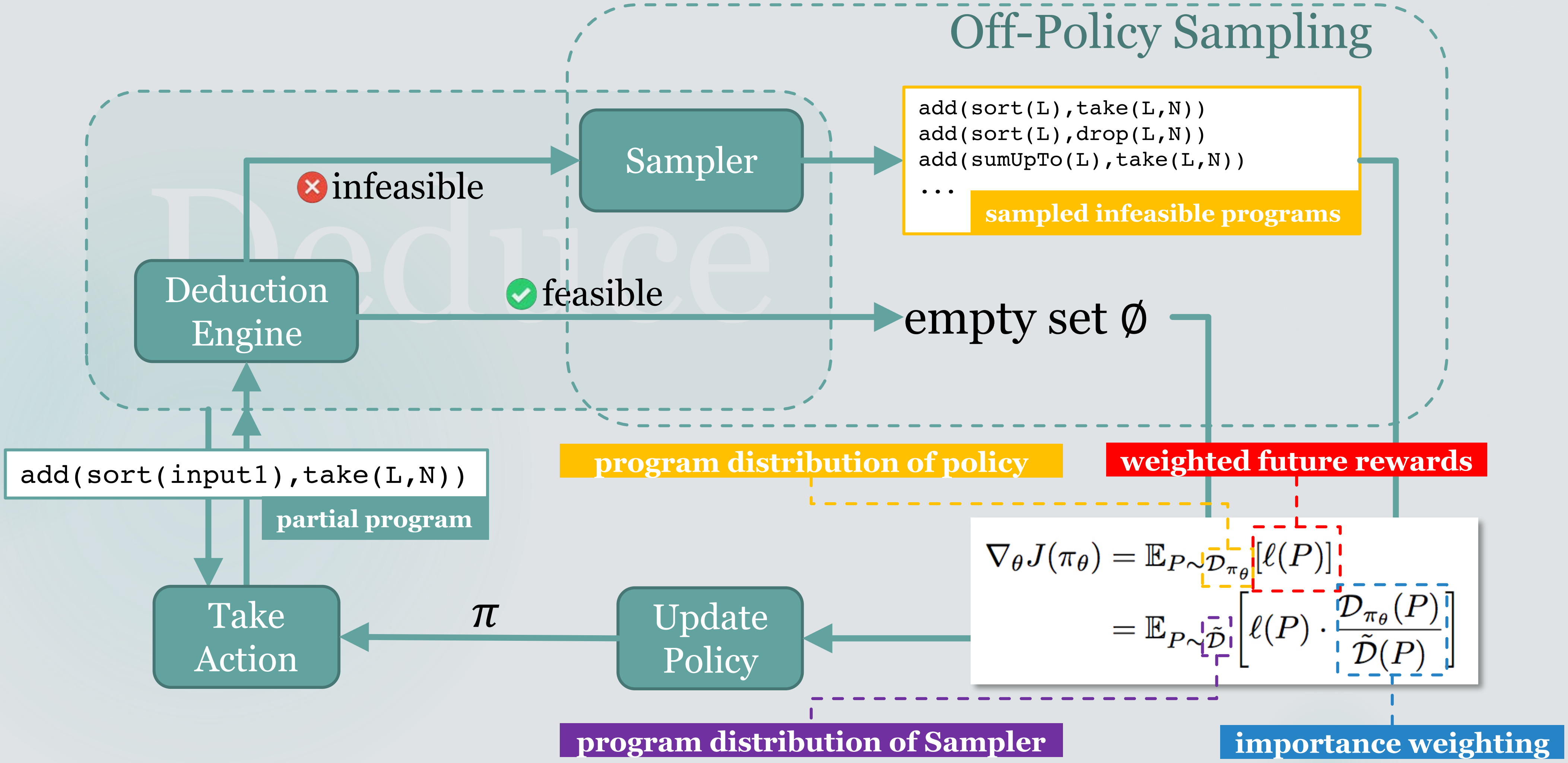


CONCORD: Synthesis Algorithm



CONCORD: Synthesis Algorithm

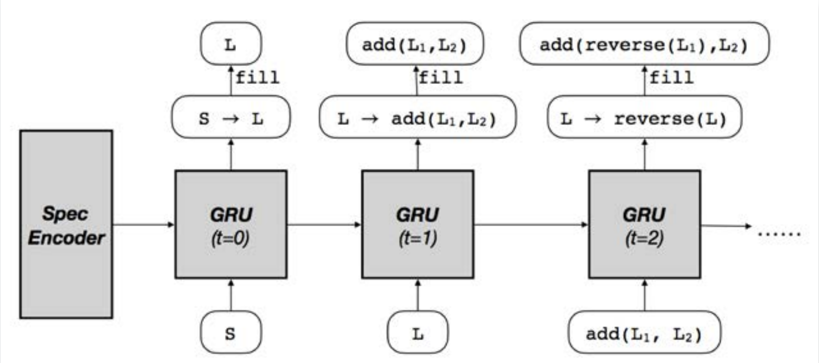
(Deduction-Guided Reinforcement Learning)



Evaluations

(Research Questions and Experiment Settings)

- Research Questions:
  - How does Concord compare against existing synthesis tools?
  - How effective is the off-policy RL algorithm compared to standard policy gradient?
- Experiment Settings
  - Deduction Engine: NEO's (Feng et al. 2018) deduction engine
  - Policy: Gated Recurrent Unit (GRU)
  - Benchmarks: DEEPCODER benchmarks used in NEO
    - 100 challenging list processing problems
  - Comparison between:
    - NEO (Feng et al. 2018)
    - DEEPCODER (Balog et al. 2017)



The architecture of the policy network used

Evaluations

(Experiment Results and Analysis)

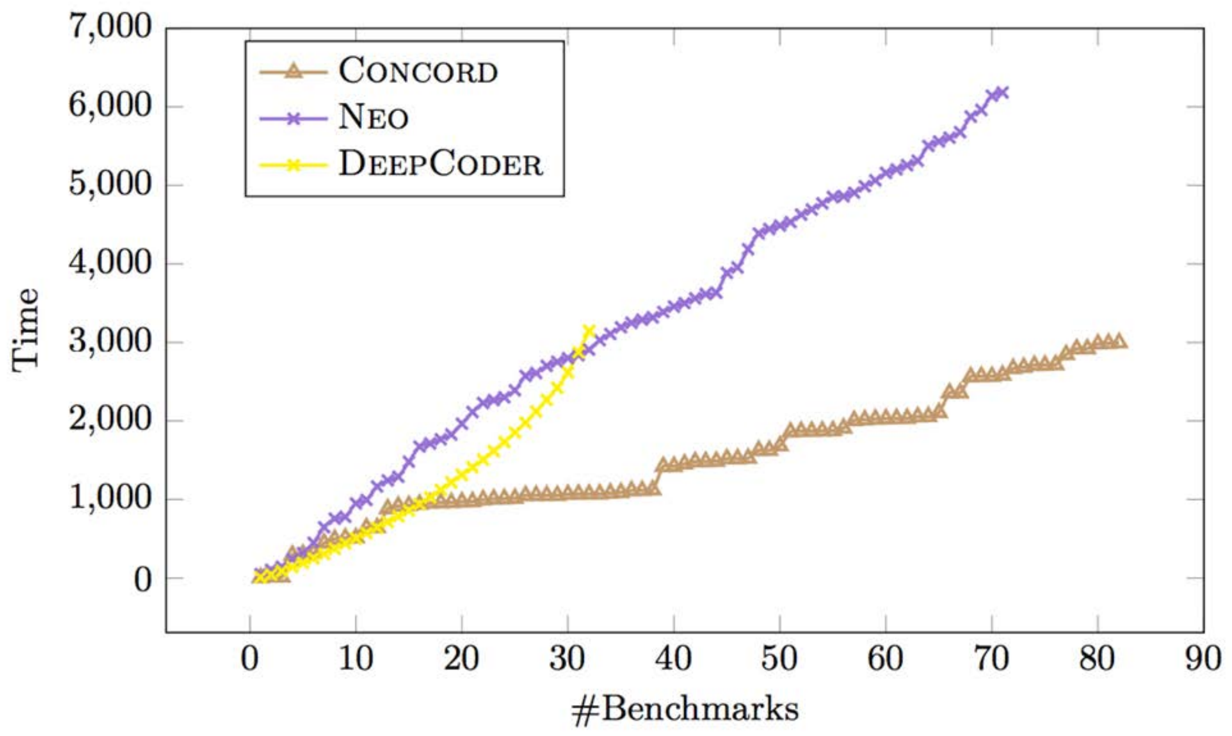


Fig. 5. Comparison between CONCORD, NEO, and DEEPCODER

tool	solved	time
CONCORD	**82%**	**36s**
NEO	71%	99s
DEEPCODER	32%	205s

tool	solved	speedup over NEO
CONCORD	82%	8.71x
CONCORD (StandardPG)	65%	2.88x

- Concord tightly couples statistical and deductive reasoning based on reinforcement learning.
- The off-policy reinforcement learning technique is effective.

Take-Away Message

Thank you!
Questions?