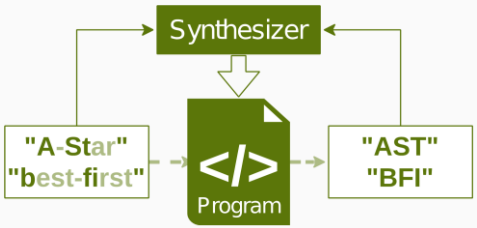
Program Synthesis with A* Search

A more robust search for more robust programs

Author: B. Jenneboer
Supervisor: S. Dumančić

1. Program Synthesis

- Automatic generation of programs from examples:



- How? **Enumerate** all programs of a language and search for one that works
- Problem: The **search space is enormous**

2. Brute [*]

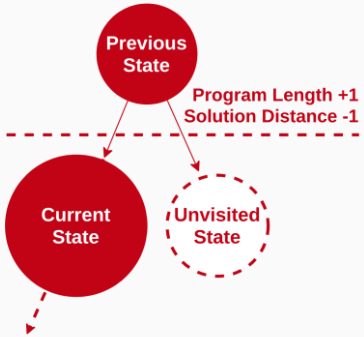
- Solution:
  - Problem-specific **custom languages** to keep the language small:

```
MoveRight
Drop
LoopWhile(IsCapital
          MoveRight)
MakeCapital
MoveRight
LoopWhile(NotAtEnd
          Drop)
```

  - **Limit** the size of **if/while** bodies
  - Traverse the search space in a smart order: Estimate the remaining **distance to a solution**
- Challenge: Brute easily becomes tempted to search **local optima** and doesn't recover within reasonable time

3. A* Search

- Alternative to Brutes **best-first-search (BFS)**
- BFS only minimizes solution distance → the unvisited program below will be left unvisited
- A* **minimizes program length** → it checks the unvisited program, before going down another level



- Advantages:
  - Solution programs are **shorter** → often more universal / **robust**
  - Ending up in a deep local optimum (possibly) less likely
- Challenge: heuristics used by Brute are **inadmissible** → no guarantee for cost optimality

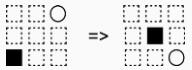
4. Research questions

- Is A* a good alternative search strategy for Brute?
  - **Q1**: Is the **predictive accuracy** of user intent better?
  - **Q2**: Can a solution be found faster on average by **avoiding local optima**?
  - **Q3**: Can we come up with more suitable (admissible) **heuristics**?

4. Method

- Re-implement Brute in an imperative language with imperative DSLs as an alternative to logic programming
- Use the same benchmarking problems Brute uses:

Robot planning



String manipulation

```
A-Star  =>  AST
^           ^
```

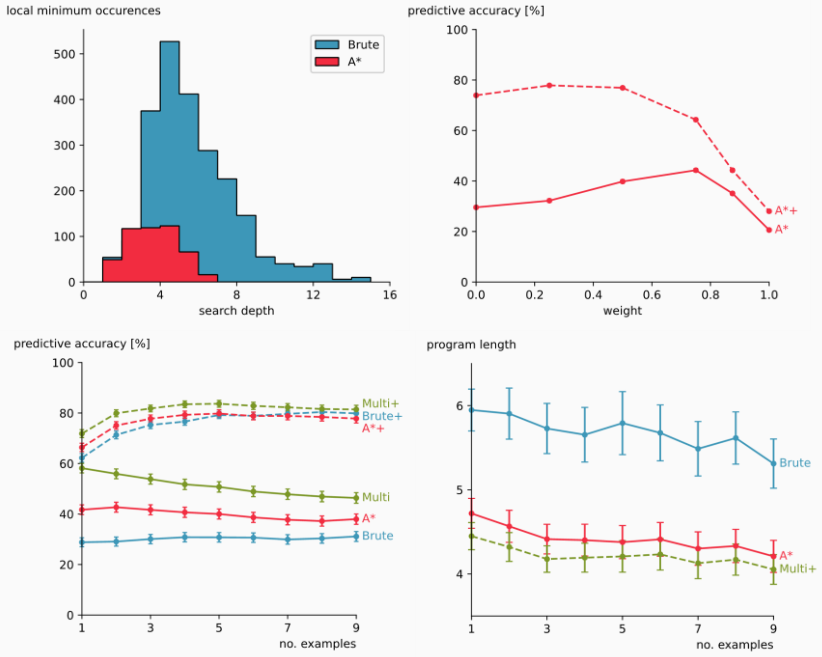
ASCII art drawing:

```
 ***   ****   ***   ****
*   *  *   *  *   *  *   *
*****  ****   *****  ****
*   *  *   *  *   *  *   *
*   *  *   *  *   *  *   *
```

- $\text{cost} = \text{weight} \times \text{program length} + (1 - \text{weight}) \times \text{distance}$

  weight $= 1/2$: A*
  $= 0$: Greedy best-first / Brute
  $= 1$: Dijkstra

5. Results



6. Conclusion/Limitations

- **Q1**: The predictive accuracy of A* is better, **provided that the heuristic is suboptimal**
- **Q2**: No, on average A* needs more time
- **Q3**: Better performing heuristics were found. Both methods benefit equally from these.

A*, Brute and other weights excel in distinct problems and could therefore complement each other.

[*] A.Cropper and S. Dumancic. Learning large logic programs by going beyond entailment. International Joint Conference on Artificial Intelligence 2020

B.Jenneboer@student.tudelft.nl