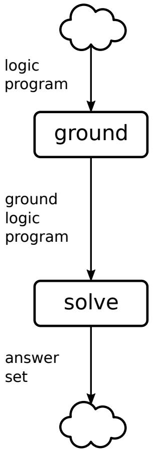
Elisa Böhl, Sarah A. Gaggl, Stefan Ellmauthaler
Logic Programming and Argumentation Group, TU Dresden, Germany

Winning Snake: Design Choices in Multi-Shot ASP

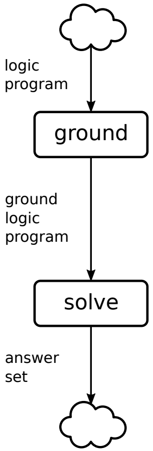
Dallas, USA, October 16th 2024

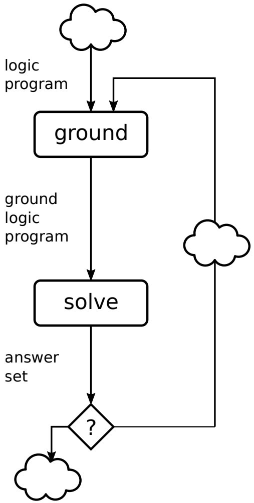
Overview

Motivation

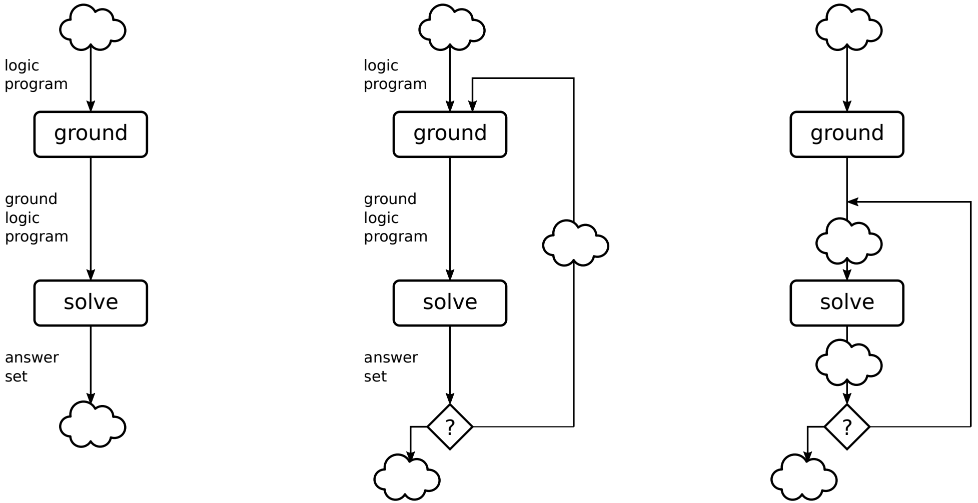

logic
program
ground
ground
logic
program
solve
answer
set

Motivation

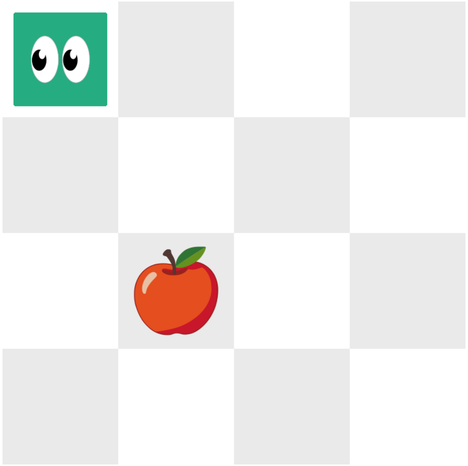

logic
program
ground
ground
logic
program
solve
answer
set


logic
program
ground
ground
logic
program
solve
answer
set
?

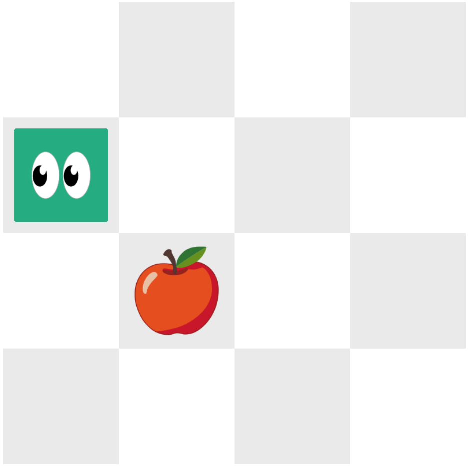
Motivation


logic
program
ground
ground
logic
program
solve
answer
set
logic
program
ground
ground
logic
program
solve
answer
set
?
ground
solve
?

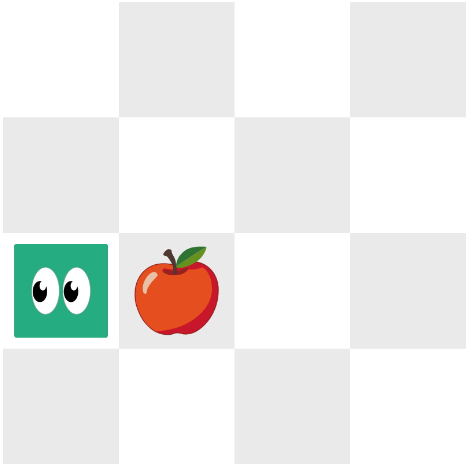
About Snakes



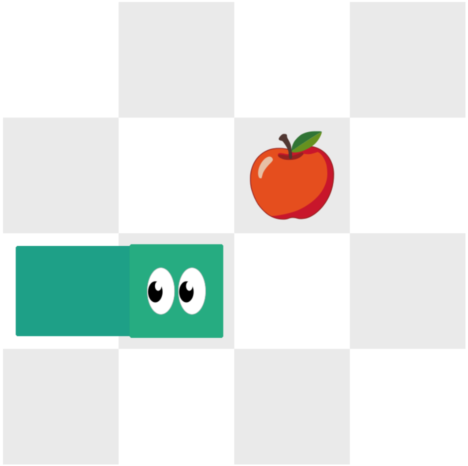
About Snakes



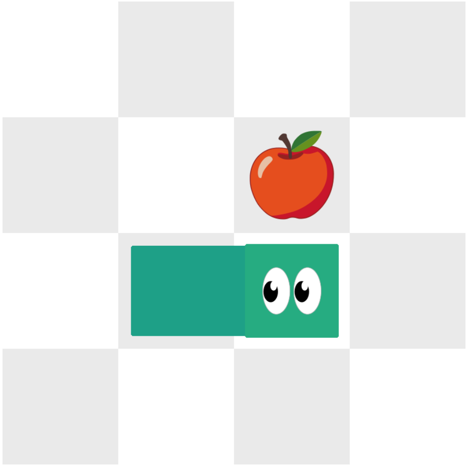
About Snakes



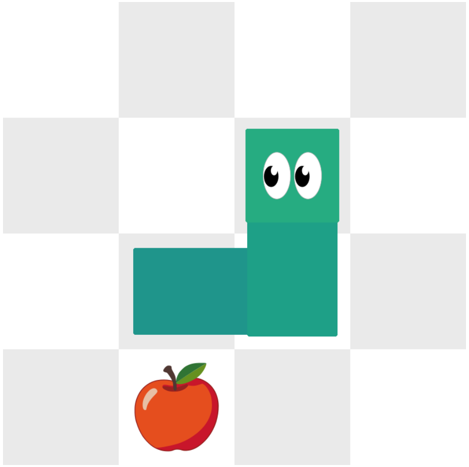
About Snakes



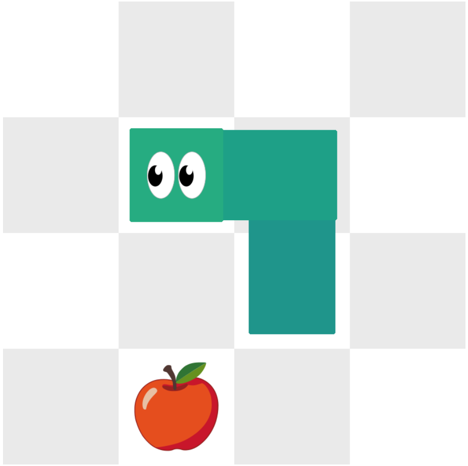
About Snakes



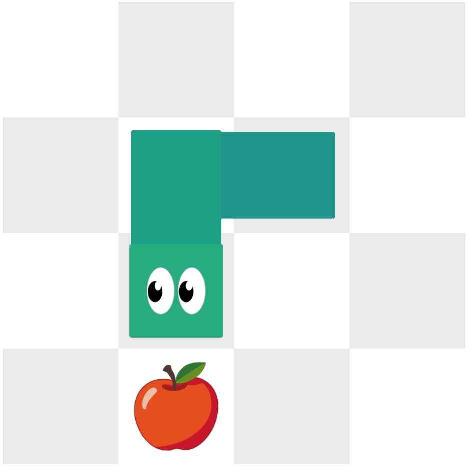
About Snakes



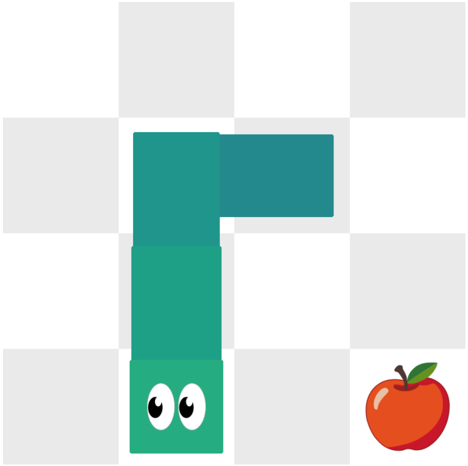
About Snakes



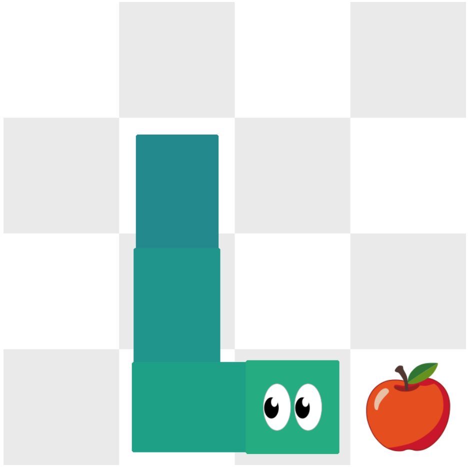
About Snakes



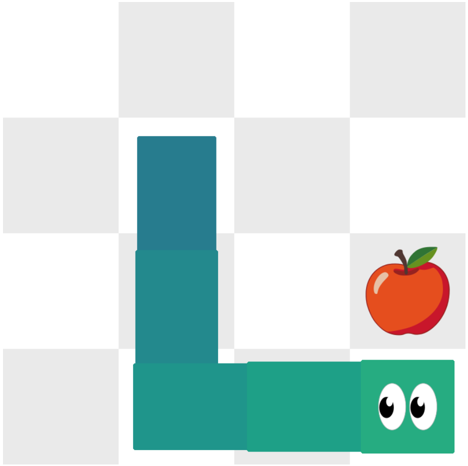
About Snakes



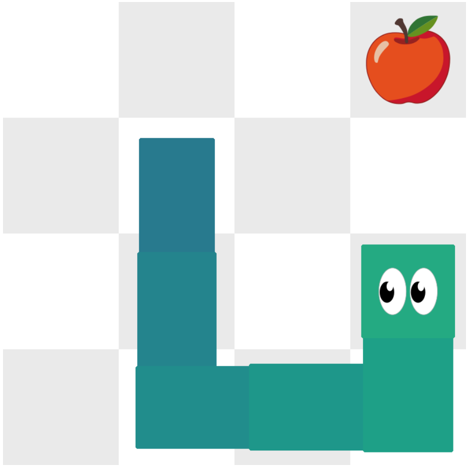
About Snakes



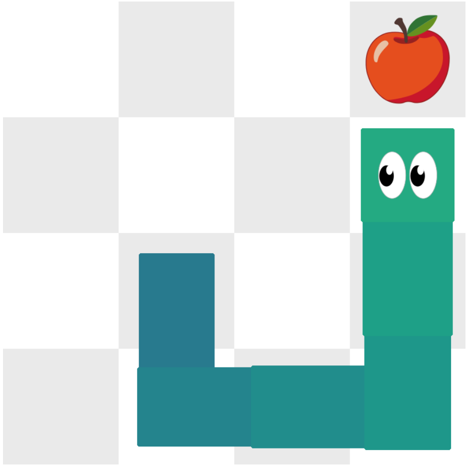
About Snakes



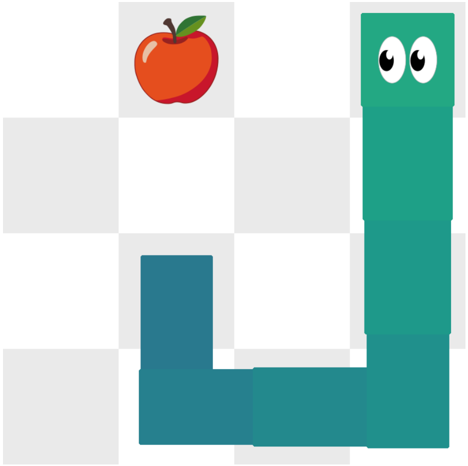
About Snakes



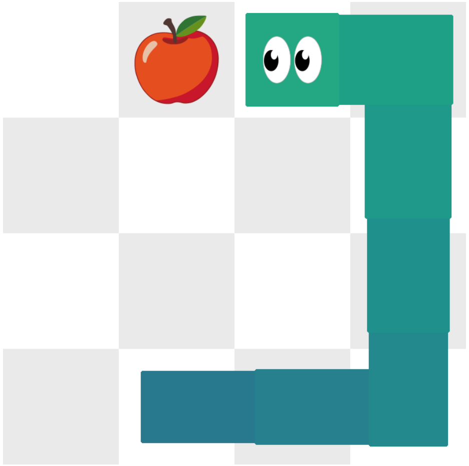
About Snakes



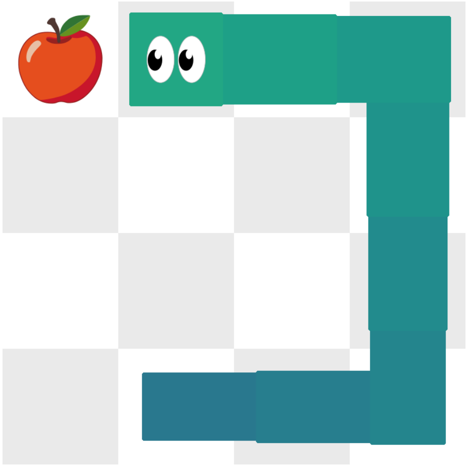
About Snakes



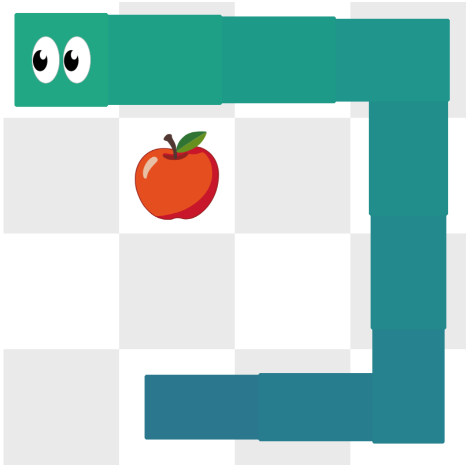
About Snakes



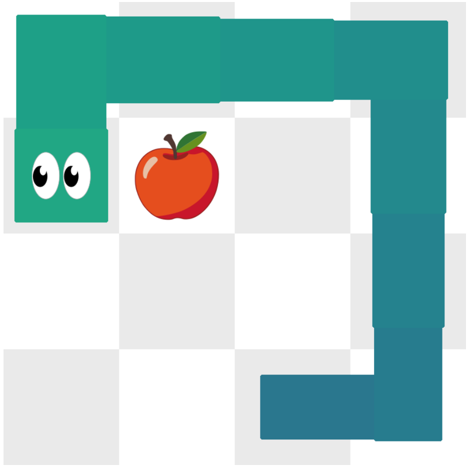
About Snakes



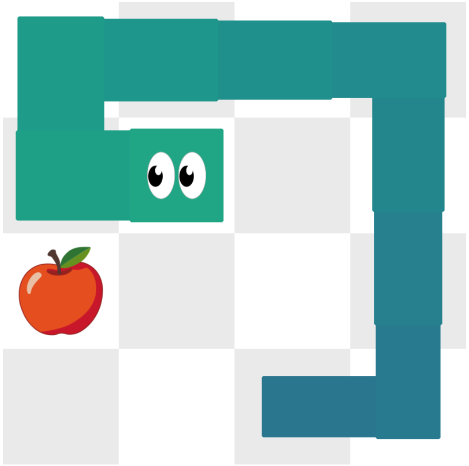
About Snakes



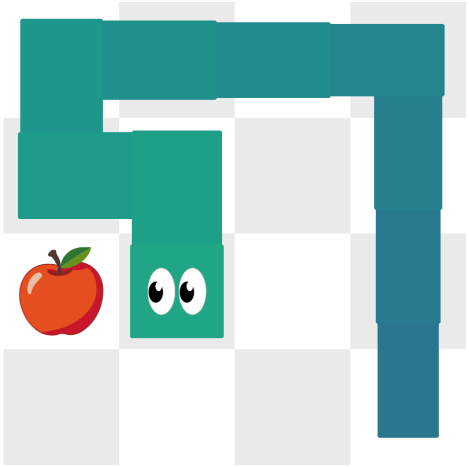
About Snakes



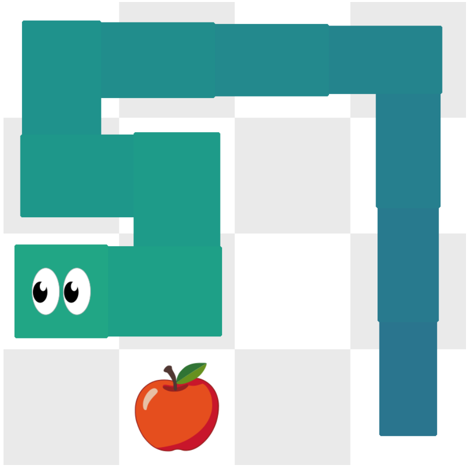
About Snakes



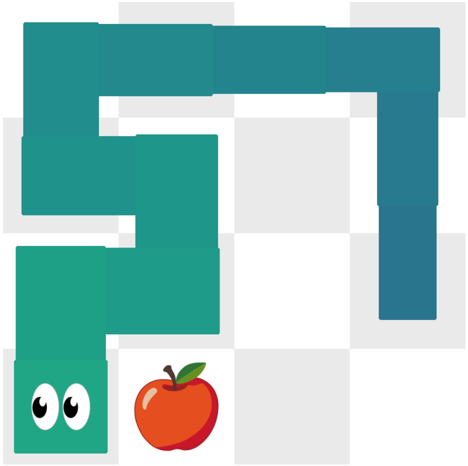
About Snakes



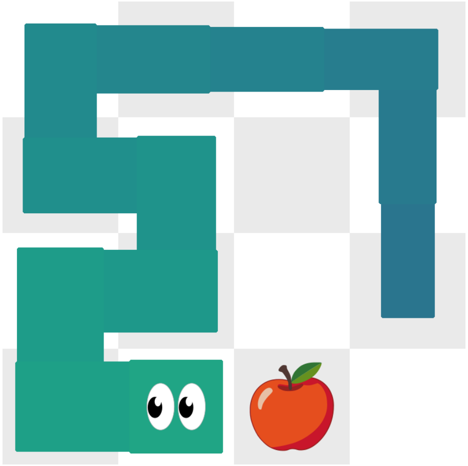
About Snakes



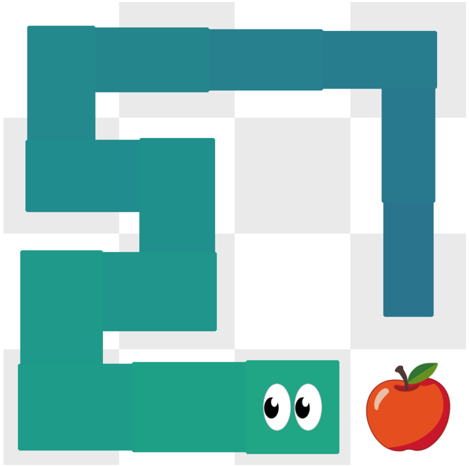
About Snakes



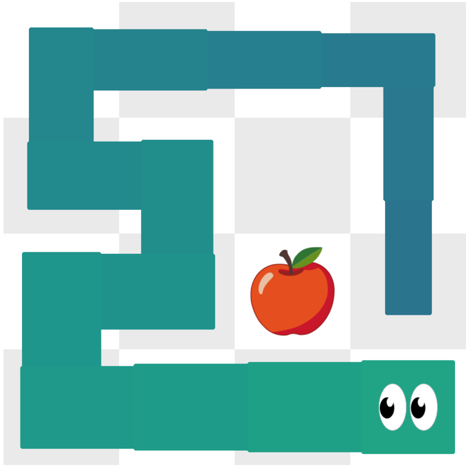
About Snakes



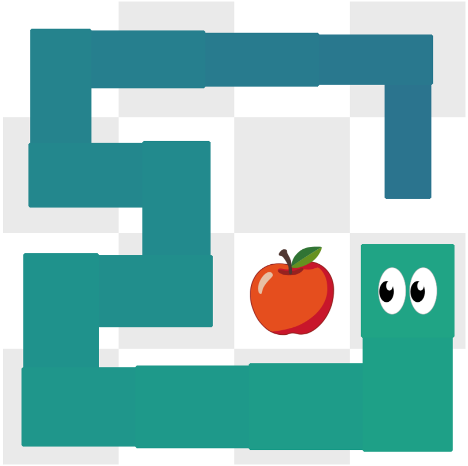
About Snakes



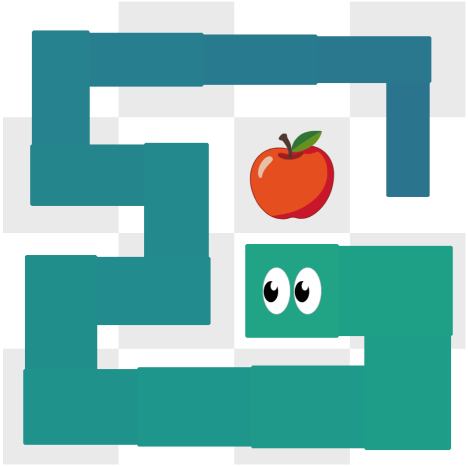
About Snakes



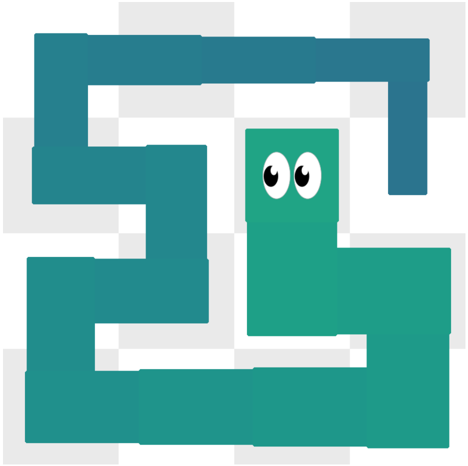
About Snakes



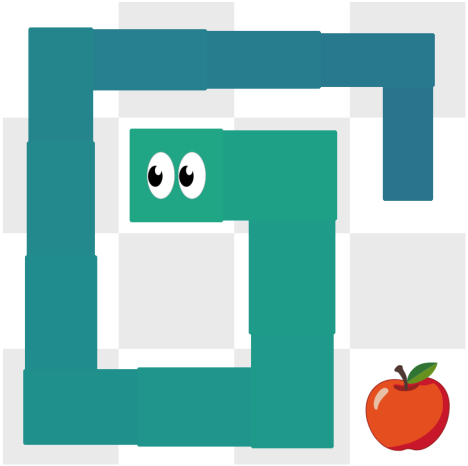
About Snakes



About Snakes



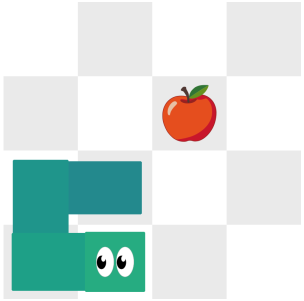
About Snakes



Snakes - Keypoints

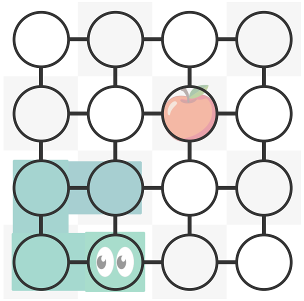
- Iterative setting
- Path from head to apple
- Solving one iteration $\not\rightarrow$ solving next iteration

Strategies



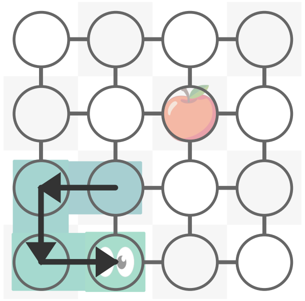
Strategies

- Interpretation as Grid Graph



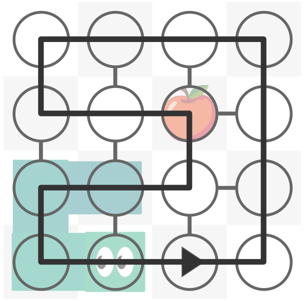
Strategies

- Interpretation as Grid Graph
- Snake placement



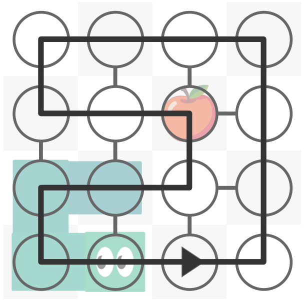
Strategies

- Interpretation as Grid Graph
- Snake placement
- Hamiltonian Cycle



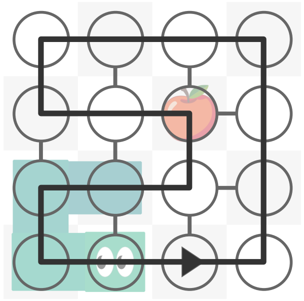
Strategies

- Interpretation as Grid Graph
- Snake placement
- Hamiltonian Cycle NP



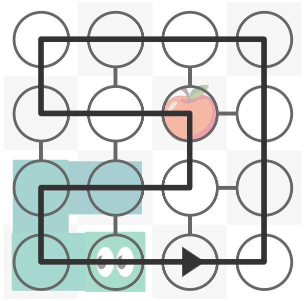
Strategies

- Interpretation as Grid Graph
- Snake placement
- Hamiltonian Cycle NP / P



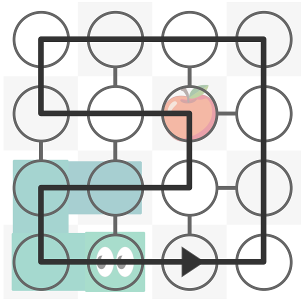
Strategies

- Interpretation as Grid Graph
- Snake placement
- Hamiltonian Cycle NP / P



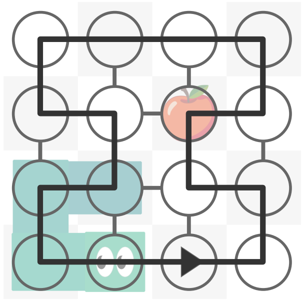
Strategies

- Interpretation as Grid Graph
- Snake placement
- Hamiltonian Cycle NP / P
- Minimize Step count NP^{NP} (conservative Strategy)



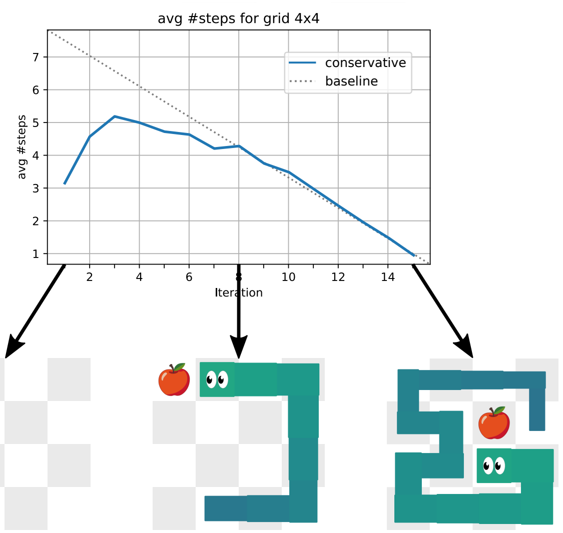
Strategies

- Interpretation as Grid Graph
- Snake placement
- Hamiltonian Cycle NP / P
- Minimize Step count NP^{NP} (conservative Strategy)



Strategies

- Interpretation as Grid Graph
- Snake placement
- Hamiltonian Cycle NP / P
- Minimize Step count NP^{NP} (conservative Strategy)



Why Snakes?

- Popular, easy to grasp yet hard
- Problem class well suited for ASP
- Iterative setting (multi-shot ASP)
- Minimalistic ASP-Encoding

Snake logic program class

Input: grid dimension n × m, position of head (head/1) and apple (apple/1)

```
field((X,Y)) :- X=1..n, Y=1..m.
conn((X,Y1),(X,Y2)) :- |Y1-Y2|=1, field((X,Y1)), field((X,Y2)).
conn((X1,Y),(X2,Y)) :- |X1-X2|=1, field((X1,Y)), field((X2,Y)).

1 { next(XY,XY') : field(XY ), conn(XY,XY') } 1 :- field(XY').
1 { next(XY,XY') : field(XY'), conn(XY,XY') } 1 :- field(XY ).

path(XY) :- field(XY), head(XY).
path(Next) :- path(XY), next(XY,Next).
:- field(XY), not path(XY).

mark(XY) :- field(XY), head(XY).
mark(Next) :- mark(XY), next(XY,Next), not apple(XY).
#minimize{ 1,XY : mark(XY) }.
```

Answer Set Programming

- Logic programming under stable model semantics
- Established problem solving paradigm
- Advanced programming techniques required
- Specific mechanisms for `clingo`

Implementation

$$\left.\begin{array}{l}\text{ad hoc}\\ \text{preground}\\ \text{assume}\\ \text{nogoods}\end{array}\right\}\ \text{multi-shot}$$

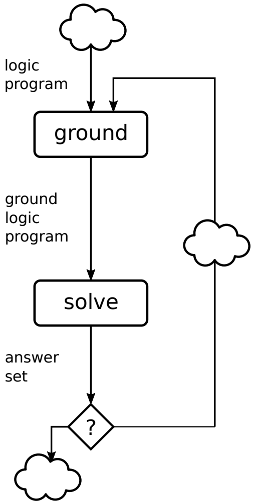
- oneshot
- ad hoc
- preground
- assume
- nogoods

Implementation

- oneshot
- ad hoc
- preground
- assume
- nogoods

$\left.\begin{array}{l}\text{ad hoc}\\\text{preground}\\\text{assume}\\\text{nogoods}\end{array}\right\}$ multi-shot

Workflow - Oneshot ⌖

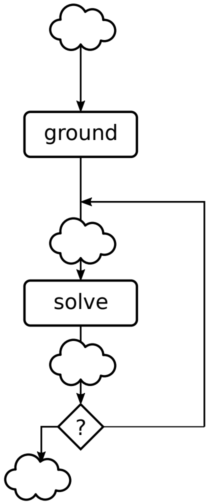


```
Input: grid dimension $n \times m$
$🐍 \leftarrow [(1,1)]$
do:
    $● \leftarrow \mathit{generate\_apple}((n,m), 🐍)$
    $\Pi \leftarrow \mathit{ground}($
            $\mathit{base\_lp}(n,m)$
            $\cup\ \{\texttt{apple(●).}\quad \texttt{head(}🐍_1\texttt{).}\}$
            $\cup\ \bigcup_{i=1..|🐍|-1}\{\texttt{next(}🐍_i\texttt{,}🐍_{i+1}\texttt{).}\})$
    $\mathit{model} \leftarrow \mathit{solve}(\Pi)$
    $\mathit{path} \leftarrow \mathit{extract\_path}(\mathit{model})$
    $🐍 \leftarrow \mathit{follow\_path}(🐍, \mathit{path}, ●)$
while $|🐍| < n \cdot m$
```

Workflow - Oneshot ⌖

- fast and straight forward implementation
- flexible design, easy debugging
- redundant steps
- suboptimal resource use

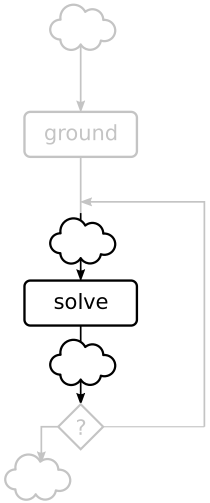
Workflow - Multi-shot



Input: grid dimension $n \times m$

```
$\Pi \leftarrow$ ground({#external apple(X) : field(X).
                         #external head(X) : field(X).}
                         $\cup$ base_lp$(n, m)$)
🐍 $\leftarrow [(1,1)]$
do :
    🍎 $\leftarrow$ generate_apple($(n, m)$, 🐍)
    $\Pi \leftarrow$ set_external($\Pi$, apple(🍎), True)
    $\Pi \leftarrow$ set_external($\Pi$, head(🐍$_1$), True)
    $\Pi$, path $\leftarrow$ retrieve($\Pi$, 🐍)
    $\Pi \leftarrow$ set_external($\Pi$, apple(🍎), False)
    $\Pi \leftarrow$ set_external($\Pi$, head(🐍$_1$), False)
    🐍 $\leftarrow$ follow_path(🐍, path, 🍎)
while |🐍| $< n \cdot m$
```

Workflow - Multi-shot



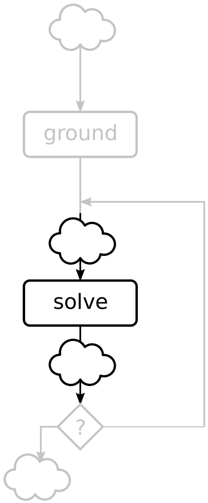
```
Input: grid dimension $n \times m$
$\Pi \leftarrow ground(\{$#external apple(X) : field(X).
                        #external head(X) : field(X).$\}$
                        $\cup\ base\_lp(n,m))$
🐍 $\leftarrow [(1,1)]$
do :
    🍎 $\leftarrow generate\_apple((n,m),$ 🐍$)$
    $\Pi \leftarrow set\_external(\Pi,$ apple(🍎), $True)$
    $\Pi \leftarrow set\_external(\Pi,$ head(🐍$_1$), $True)$
    $\Pi, path \leftarrow$ retrieve$(\Pi,$ 🐍$)$
    $\Pi \leftarrow set\_external(\Pi,$ apple(🍎), $False)$
    $\Pi \leftarrow set\_external(\Pi,$ head(🐍$_1$), $False)$
    🐍 $\leftarrow follow\_path($🐍$, path,$ 🍎$)$
while |🐍| $< n \cdot m$
```

Implementation

- oneshot
- ad hoc
- preground
- assume
- nogoods

$\left.\begin{array}{l}\text{ad hoc}\\\text{preground}\\\text{assume}\\\text{nogoods}\end{array}\right\}$ multi-shot

Workflow - Ad hoc ±



Algorithm 1: *retrieve* for ad hoc; Input: Π, 🐍; Output: Π, *path*

```
1 $\Pi \leftarrow \Pi \cup ground(\{$#external step(|🐍|).$\})$
2 for $i = 1 .. |🐍|-1$ :
3     $\Pi \leftarrow \Pi \cup ground(\{$:- step(|🐍|), not next(🐍$_i$,🐍$_{i+1}$).$\})$
4 $\Pi \leftarrow set\_external(\Pi,$ step(|🐍|)$, True)$
5 $model \leftarrow solve(\Pi)$
6 $\Pi \leftarrow release\_external(\Pi,$ step(|🐍|)$)$
7 $\Pi \leftarrow cleanup(\Pi)$
8 return $\Pi, extract\_path(model)$
```

Workflow - Ad hoc ±

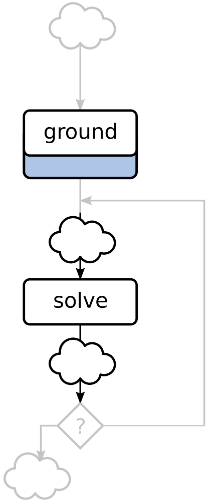
- full flexibility, introduce new atoms
- easy implementation vs. code reusing
- applied standard in current publications

Implementation

- oneshot

$$\left.\begin{array}{l}\text{ad hoc}\\ \text{preground}\\ \text{assume}\\ \text{nogoods}\end{array}\right\}\ \text{multi-shot}$$

Workflow - Preground



logic program extension:

```
#external prenext(X,Y) : connected(X,Y).
:- prenext(X,Y), not next(X,Y), connected(X,Y).
```

Algorithm 2: *retrieve* for preground; Input: Π, 🐍; Output: Π, *path*

```
1 for $i = 1..|🐍| - 1$:
2     $\Pi \leftarrow set\_external(\Pi, \texttt{prenext(}🐍_i\texttt{,}🐍_{i+1}\texttt{)}, True)$
3 $model \leftarrow solve(\Pi)$
4 for $i = 1..|🐍| - 1$:
5     $\Pi \leftarrow set\_external(\Pi, \texttt{prenext(}🐍_i\texttt{,}🐍_{i+1}\texttt{)}, False)$
6 return $\Pi, extract\_path(model)$
```

Workflow - Preground

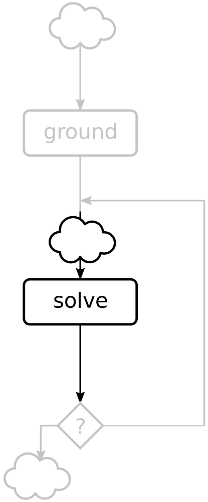
- easy interface
- complex expressions possible
- for a compact extensions
- least flexible

Implementation

- oneshot
- ad hoc
- preground
- assume
- nogoods

$\left.\begin{array}{l}\text{ad hoc}\\ \text{preground}\\ \text{assume}\\ \text{nogoods}\end{array}\right\}$ multi-shot

Workflow - Assume



Algorithm 3: retrieve for assume; Input: Π, 🐍; Output: Π, *path*

```
1 assume ← [ ]
2 for $i = 1..|🐍| - 1$:
3     assume.append((next(🐍$_i$, 🐍$_{i+1}$), True))
4 model ← solve($\Pi$, assumption = assume)
5 return $\Pi$, extract_path(model)
```

Workflow - Assume

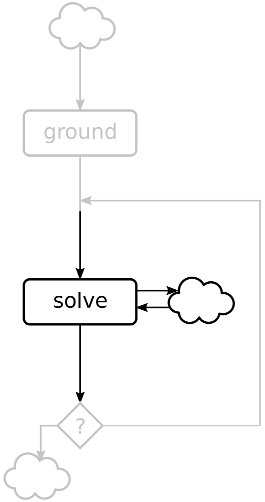
- manipulate any atom
- no reset required
- logic program stays untouched
- least expressivity
- no interface identifiers

Implementation

- oneshot
- ad hoc
- preground
- assume
- nogoods

multi-shot

Workflow - Nogoods



Algorithm 4: code snipped to add 🐍 as search restriction

Input: model *model*

```
1 ...
2 if dummy ∈ model:
3     for $i = 1..|🐍| - 1$:
4         model.context.add_clause(next(🐍_i, 🐍_{i+1}), True)
5 ...
```

Workflow - Nogoods

- manipulate any clause
- logic program stays untouched
- no interface identifiers
- complex access
- initial model problem

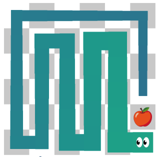
Attribute summary

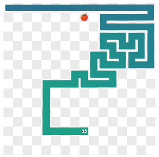
- prototyping/debugging: oneshot
- resource optimization: multi-shot

	ad hoc	preground	assume	nogood
flexibility	introduce rules	predefined	predefined	predefined
expressivity	rules	constraints	atoms	clauses
program alternation	every iteration	once	none/once	none
interface	NA	good	depends	depends
accessibility	good	good	good	hidden

Experiments - Setup

- 6 different square grid sizes ($n = m, n \in \{6, 8, 10, 12, 14, 16\}$)
- 100 repetitions per grid size and approach
- 60s timeout for solve per iteration
- average number of steps, average total time
- MacBook Pro (2017, 16 GB RAM, Intel Core i7, 2.8 GHz)
- clingo v. 5.4.0, python v. 3.7.4
- ASP-Chef for prototyping, `clingraph` for graphics
- github.com/elbo4u/asp-snake-ms





Experiments - Evaluation

average total Time (grounding and solving) in seconds

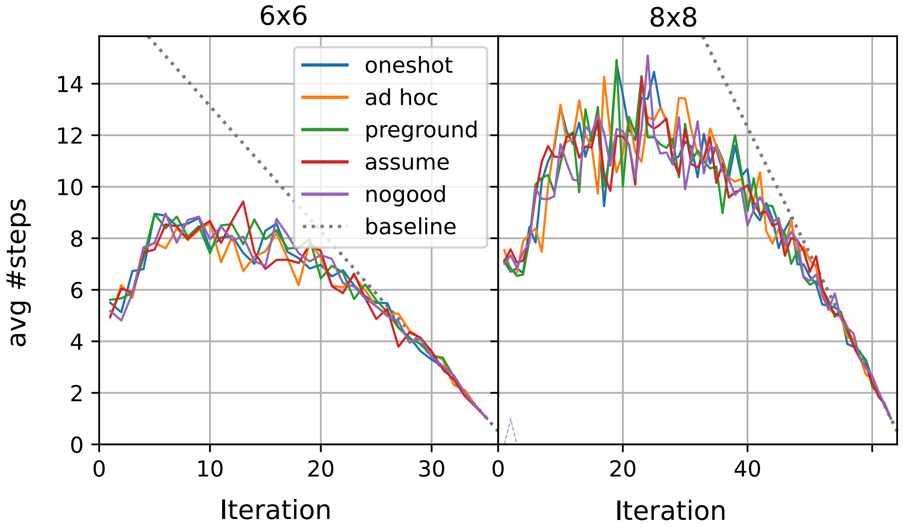
	$n \times m$	6×6	8×8	10×10	12×12	14×14	16×16
	one-shot	0.159	**2.28**	**71.83**	621	2216	4359
	ad hoc	0.066	3.42	90.07	674	1966	3869
time	preground	0.060	3.32	94.71	**620**	1978	3870
	assume	**0.059**	4.78	97.48	628	**1944**	**3853**
	nogood	0.061	3.16	94.70	702	1951	3877

Experiments - Evaluation

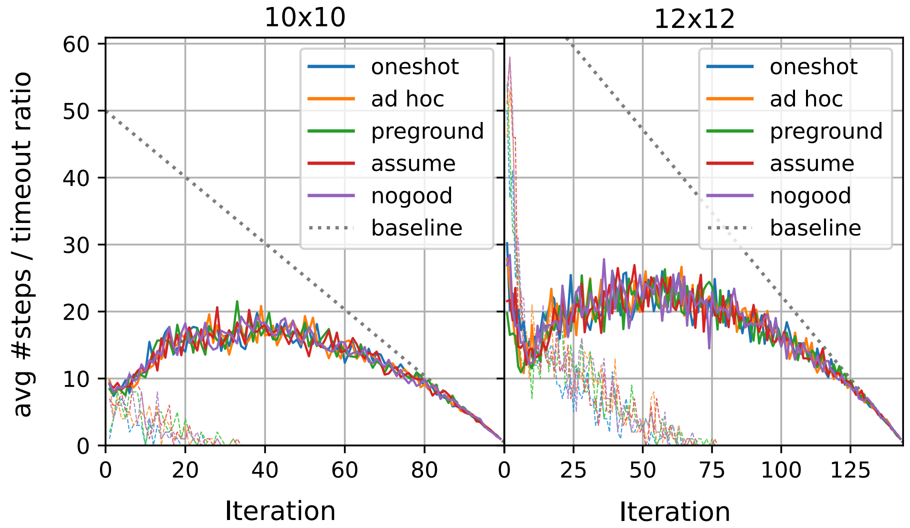
average total Time (grounding and solving) in seconds

	$n \times m$	6×6	8×8	10×10	12×12	14×14	16×16
	one-shot	0.159	**2.28**	**71.83**	621	2216	4359
	ad hoc	0.066	3.42	90.07	674	1966	3869
time	preground	0.060	3.32	94.71	**620**	1978	3870
	assume	**0.059**	4.78	97.48	628	**1944**	**3853**
	nogood	0.061	3.16	94.70	702	1951	3877
	one-shot	213	576	1235	2441	<u>5519</u>	<u>10157</u>
	ad hoc	208	572	1226	2411	4582	7445
steps	preground	216	563	1236	2374	4540	7482
	assume	210	563	1234	2396	4508	7540
	nogood	212	559	1240	2428	4580	7523

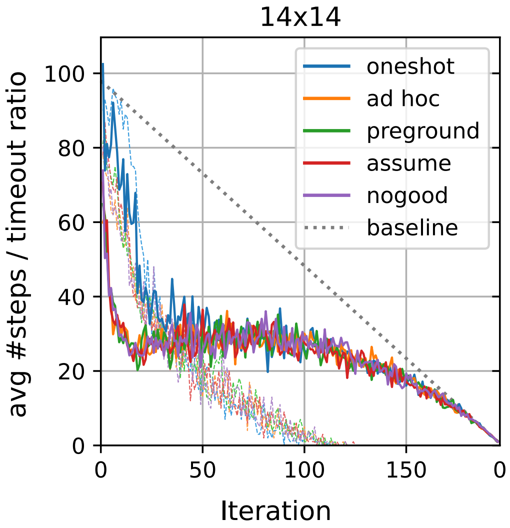
Experiments - Evaluation - Steps



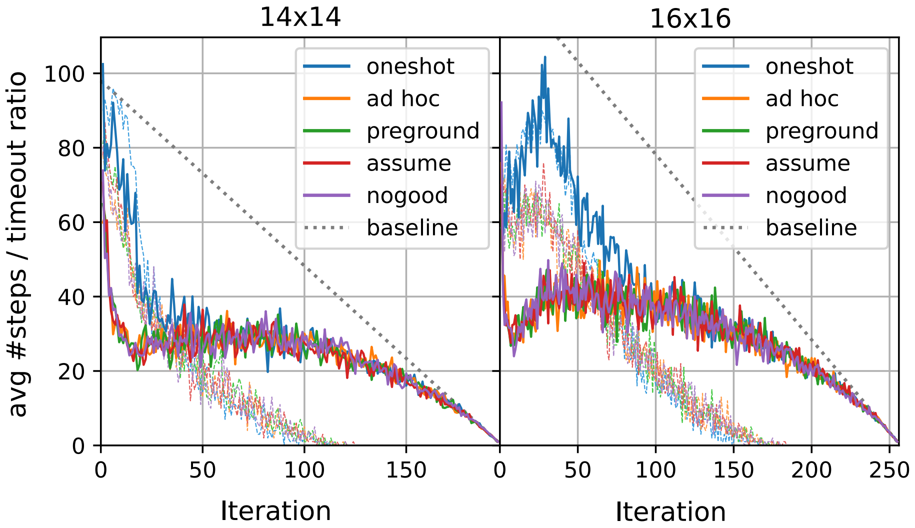
Experiments - Evaluation - Steps



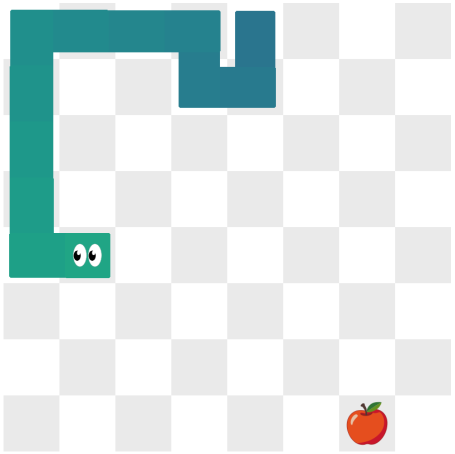
Experiments - Evaluation - Steps



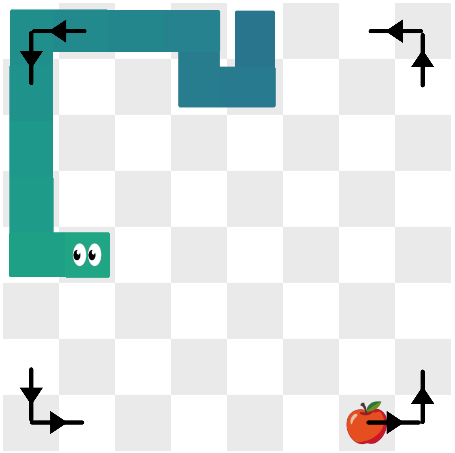
Experiments - Evaluation - Steps



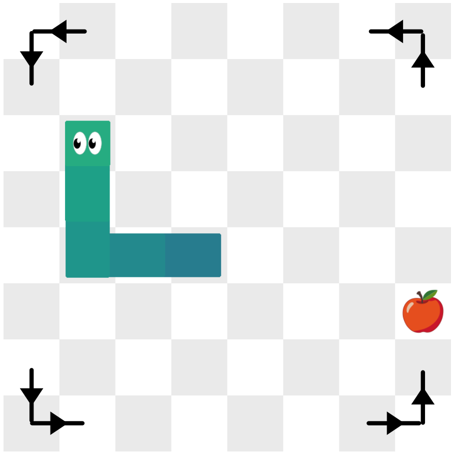
Experiments - Evaluation - Nogoods



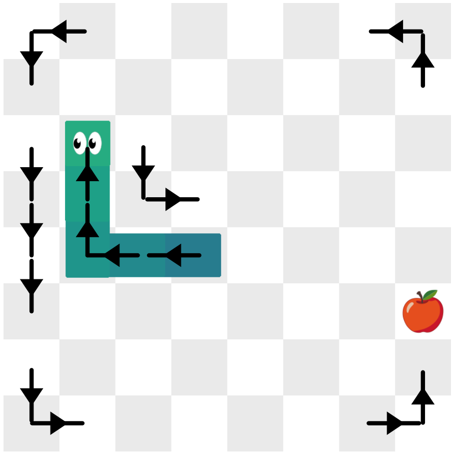
Experiments - Evaluation - Nogoods



Experiments - Evaluation - Nogoods



Experiments - Evaluation - Nogoods



Conclusion and Future Work

Conclusion

- Different approaches to implement multi-shot in `clingo`
- Snakes as suited showcase benchmark
- Multi-shot as reasonable technique for limited resources

Conclusion and Future Work

Conclusion

- Different approaches to implement multi-shot in `clingo`
- Snakes as suited showcase benchmark
- Multi-shot as reasonable technique for limited resources

Future Work

- Different benchmark to highlight differences
- More sophisticated strategies
- Strategies to handle arbitrary snakes
- DLV Implementation

Conclusion and Future Work

Conclusion

- Different approaches to implement multi-shot in `clingo`
- Snakes as suited showcase benchmark
- Multi-shot as reasonable technique for limited resources

Future Work

- Different benchmark to highlight differences
- More sophisticated strategies
- Strategies to handle arbitrary snakes
- DLV Implementation

Conclusion and Future Work

Conclusion

- Different approaches to implement multi-shot in `clingo`
- Snakes as suited showcase benchmark
- Multi-shot as reasonable technique for limited resources

Future Work

- Different benchmark to highlight differences
- More sophisticated strategies
- Strategies to handle arbitrary snakes
- DLV Implementation

Conclusion and Future Work

Conclusion

- Different approaches to implement multi-shot in `clingo`
- Snakes as suited showcase benchmark
- Multi-shot as reasonable technique for limited resources

Future Work

- Different benchmark to highlight differences
- More sophisticated strategies
- Strategies to handle arbitrary snakes
- DLV Implementation

Conclusion and Future Work

Conclusion

- Different approaches to implement multi-shot in `clingo`
- Snakes as suited showcase benchmark
- Multi-shot as reasonable technique for limited resources

Future Work

- Different benchmark to highlight differences
- More sophisticated strategies
- Strategies to handle arbitrary snakes
- DLV Implementation

Conclusion and Future Work

Conclusion

- Different approaches to implement multi-shot in `clingo`
- Snakes as suited showcase benchmark
- Multi-shot as reasonable technique for limited resources

Future Work

- Different benchmark to highlight differences
- More sophisticated strategies
- Strategies to handle arbitrary snakes
- DLV Implementation

Conclusion and Future Work

Conclusion

- Different approaches to implement multi-shot in `clingo`
- Snakes as suited showcase benchmark
- Multi-shot as reasonable technique for limited resources

Future Work

- Different benchmark to highlight differences
- More sophisticated strategies
- Strategies to handle arbitrary snakes
- DLV Implementation

Conclusion and Future Work

Conclusion

- Different approaches to implement multi-shot in `clingo`
- Snakes as suited showcase benchmark
- Multi-shot as reasonable technique for limited resources

Future Work

- Different benchmark to highlight differences
- More sophisticated strategies
- Strategies to handle arbitrary snakes
- DLV Implementation

Conclusion and Future Work

Conclusion

- Different approaches to implement multi-shot in `clingo`
- Snakes as suited showcase benchmark
- Multi-shot as reasonable technique for limited resources

Future Work

- Different benchmark to highlight differences
- More sophisticated strategies
- Strategies to handle arbitrary snakes
- DLV Implementation

Conclusion and Future Work

Conclusion

- Different approaches to implement multi-shot in `clingo`
- Snakes as suited showcase benchmark
- Multi-shot as reasonable technique for limited resources

Future Work

- Different benchmark to highlight differences
- More sophisticated strategies
- Strategies to handle arbitrary snakes
- DLV Implementation

Conclusion and Future Work

Conclusion

- Different approaches to implement multi-shot in `clingo`
- Snakes as suited showcase benchmark
- Multi-shot as reasonable technique for limited resources

Future Work

- Different benchmark to highlight differences
- More sophisticated strategies
- Strategies to handle arbitrary snakes
- DLV Implementation

Conclusion and Future Work

Conclusion

- Different approaches to implement multi-shot in `clingo`
- Snakes as suited showcase benchmark
- Multi-shot as reasonable technique for limited resources

Future Work

- Different benchmark to highlight differences
- More sophisticated strategies
- Strategies to handle arbitrary snakes
- DLV Implementation

Conclusion and Future Work

Conclusion

- Different approaches to implement multi-shot in `clingo`
- Snakes as suited showcase benchmark
- Multi-shot as reasonable technique for limited resources

Future Work

- Different benchmark to highlight differences
- More sophisticated strategies
- Strategies to handle arbitrary snakes
- DLV Implementation