TOWARDS THE USAGE OF WINDOW COUNTING CONSTRAINTS IN THE SYNTHESIS OF REACTIVE SYSTEMS TO REDUCE STATE SPACE EXPLOSION

Linda Feeken, Martin Fränzle

GandALF 2024, 20.06.2024

DLR

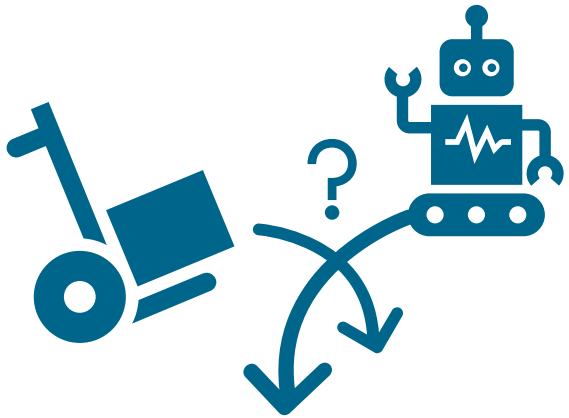
TOWARDS THE USAGE OF WINDOW COUNTING CONSTRAINTS IN THE **SYNTHESIS OF REACTIVE SYSTEMS** TO REDUCE STATE SPACE EXPLOSION

Linda Feeken, German Aerospace Center (DLR), 20.06.2024

Synthesis of Reactive Systems

Short Introduction

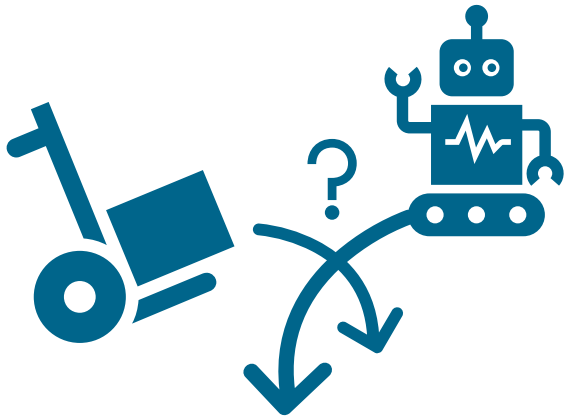
System interacts with environment



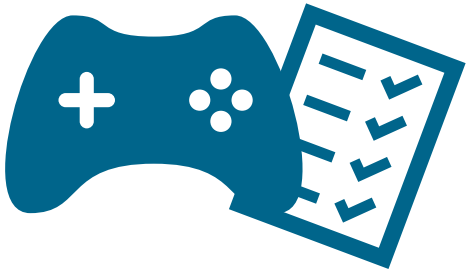
Synthesis of Reactive Systems

Short Introduction

System interacts with environment



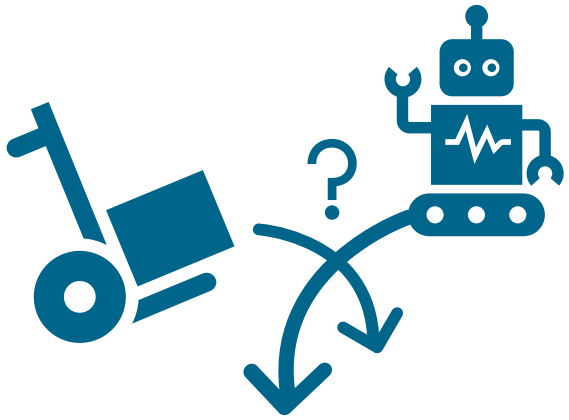
Strategies compliant with system specification



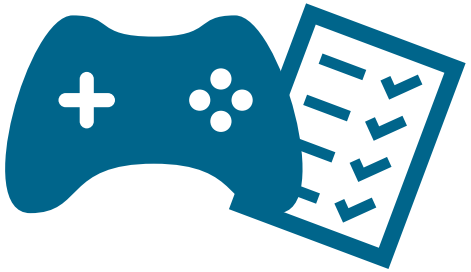
Synthesis of Reactive Systems

Short Introduction

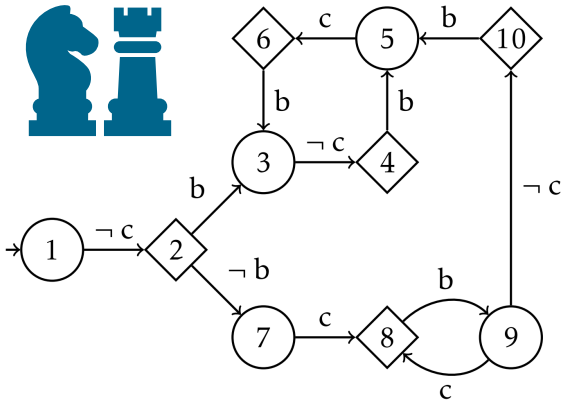
System interacts with environment



Strategies compliant with system specification



Game-theoretic approach



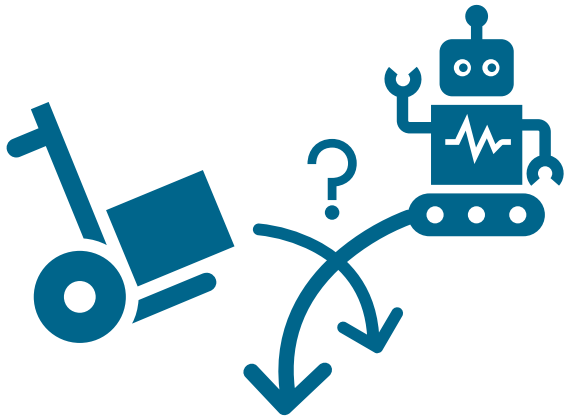
Synthesis of Reactive Systems

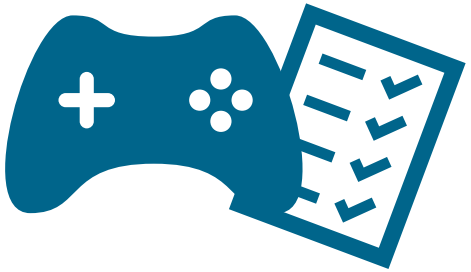
Short Introduction

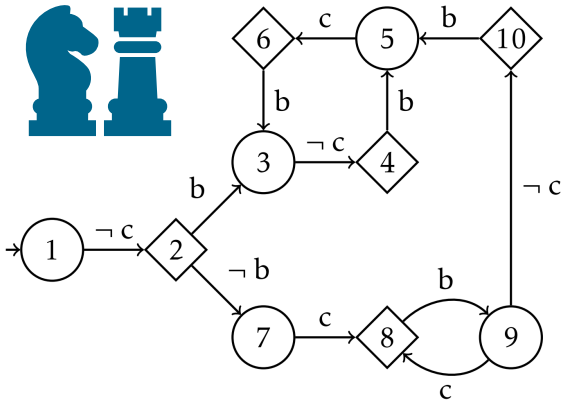
System interacts with environment

Strategies compliant with system specification

Game-theoretic approach







Here:

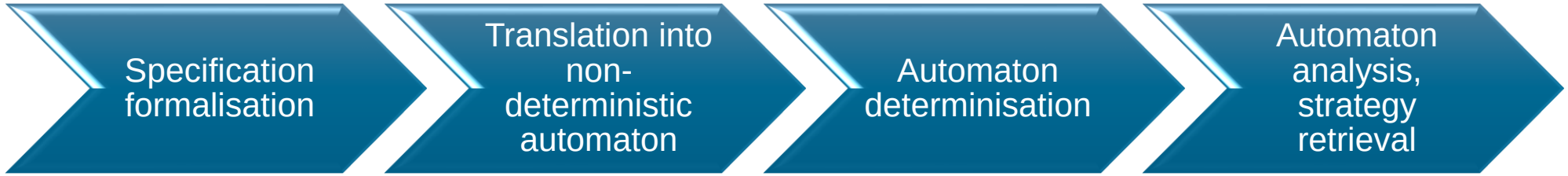
- Two-player finite-state game with infinite plays
- Specification given as combination of window counting constraints and explicit game graph

TOWARDS THE USAGE OF WINDOW COUNTING CONSTRAINTS IN THE SYNTHESIS OF REACTIVE SYSTEMS TO REDUCE **STATE SPACE EXPLOSION**

Linda Feeken, German Aerospace Center (DLR), 20.06.2024

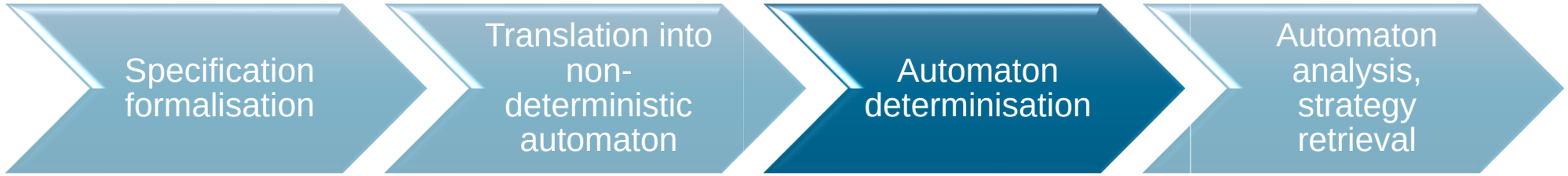
State Space Explosion

Standard synthesis procedure overview



State Space Explosion

Standard synthesis procedure overview



Expensive procedure!
(computational time, memory)
Resulting automaton is significantly bigger than specification! State space explosion!

State Space Explosion

Related Work to deal with state space explosion

Synthesis Input: Specification

Synthesis Procedure

Synthesis Output: Strategy

State Space Explosion

Related Work to deal with state space explosion

Synthesis Input: Specification

Synthesis Procedure

Synthesis Output: Strategy

Restriction to specifications of a special form, e.g.:

- GR(1) [1]
- Safety LTL [2]

[1] Piterman, Nir, Amir Pnueli, and Yaniv Sa'ar. "Synthesis of reactive (1) designs." *Verification, Model Checking, and Abstract Interpretation: 7th International Conference, VMCAI 2006, Charleston, SC, USA, January 8-10, 2006. Proceedings 7*. Springer Berlin Heidelberg, 2006.

[2] Zhu, Shufang, et al. "A symbolic approach to safety LTL synthesis." *Hardware and Software: Verification and Testing: 13th International Haifa Verification Conference, HVC 2017, Haifa, Israel, November 13-15, 2017, Proceedings 13*. Springer International Publishing, 2017.

State Space Explosion

Related Work to deal with state space explosion

Synthesis Input: Specification

Synthesis Procedure

Synthesis Output: Strategy

Restriction to specifications of a special form, e.g.:
- GR(1)
- Safety LTL

Tackling the internal representation of the problem, e.g.:
- Safraless approaches, e.g. [3]
- Symbolic approaches, e.g. [4]

[3] Kupferman, Orna, and Moshe Y. Vardi. "Safraless decision procedures." *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. IEEE, 2005.

[4] Filiot, Emmanuel, Naiyong Jin, and Jean-François Raskin. "An antichain algorithm for LTL realizability." *Computer Aided Verification: 21st International Conference, CAV 2009, Grenoble, France, June 26-July 2, 2009. Proceedings 21*. Springer Berlin Heidelberg, 2009.

State Space Explosion

Related Work to deal with state space explosion

Synthesis Input: Specification

Synthesis Procedure

Synthesis Output: Strategy

Restriction to specifications of a special form, e.g.:
- GR(1)
- Safety LTL

Tackling the internal representation of the problem, e.g.:
- „Safraless“ approaches
- Symbolic approaches

Restriction of the strategy size, e.g.:
- Bounded synthesis, e.g. [5]
- Lazy synthesis [6]

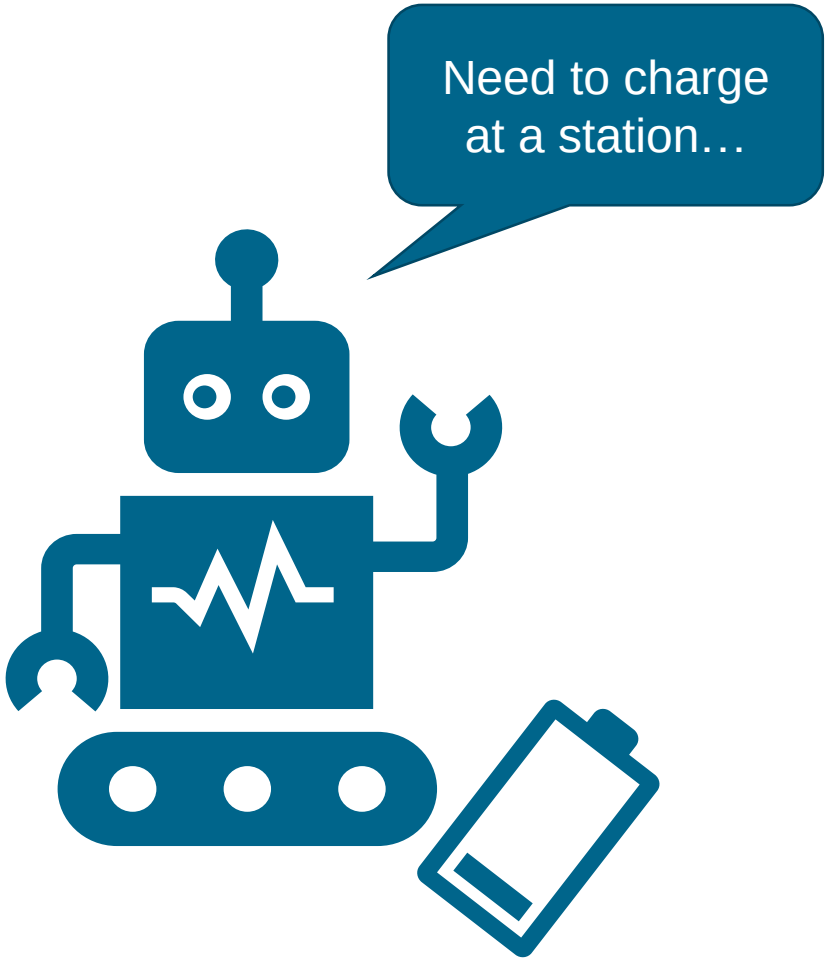
[5] Schewe, Sven, and Bernd Finkbeiner. "Bounded synthesis." *International symposium on automated technology for verification and analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[6] Finkbeiner, Bernd, and Swen Jacobs. "Lazy synthesis." *International Workshop on Verification, Model Checking, and Abstract Interpretation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

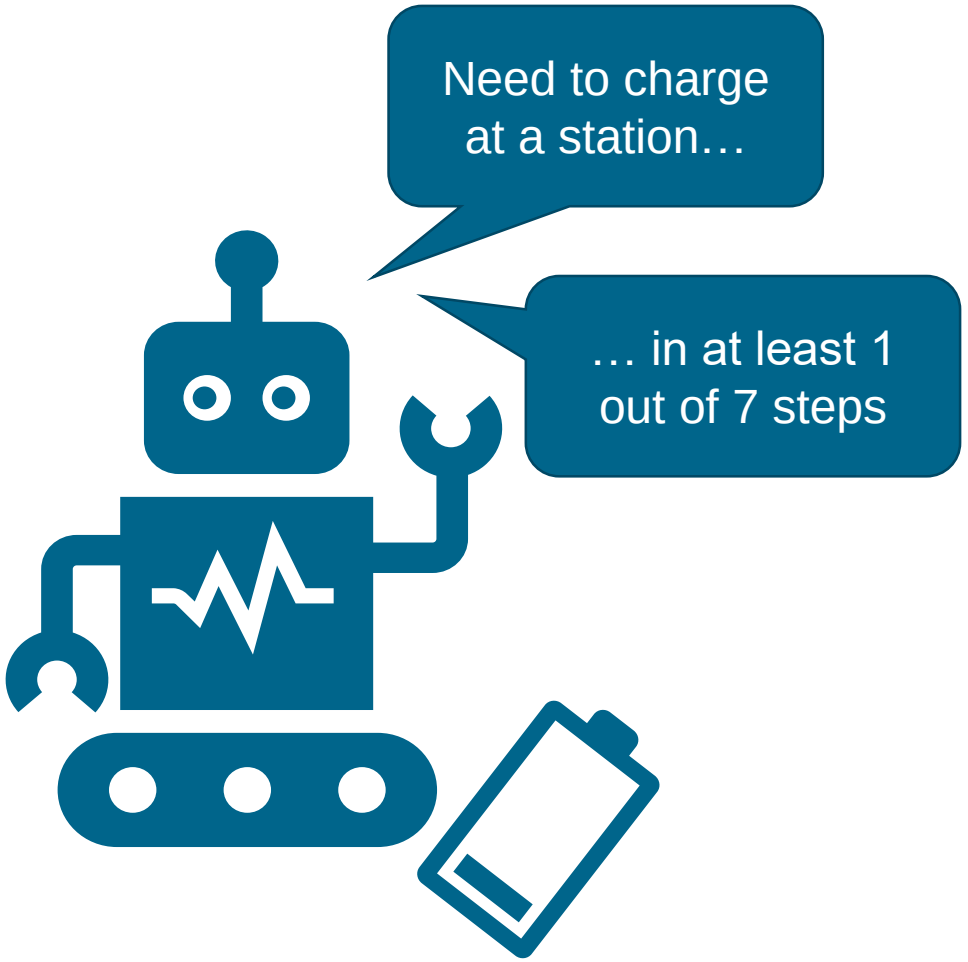
TOWARDS THE USAGE OF **WINDOW COUNTING CONSTRAINTS** IN THE SYNTHESIS OF REACTIVE SYSTEMS TO REDUCE STATE SPACE EXPLOSION

Linda Feeken, German Aerospace Center (DLR), 20.06.2024

Window Counting Constraints

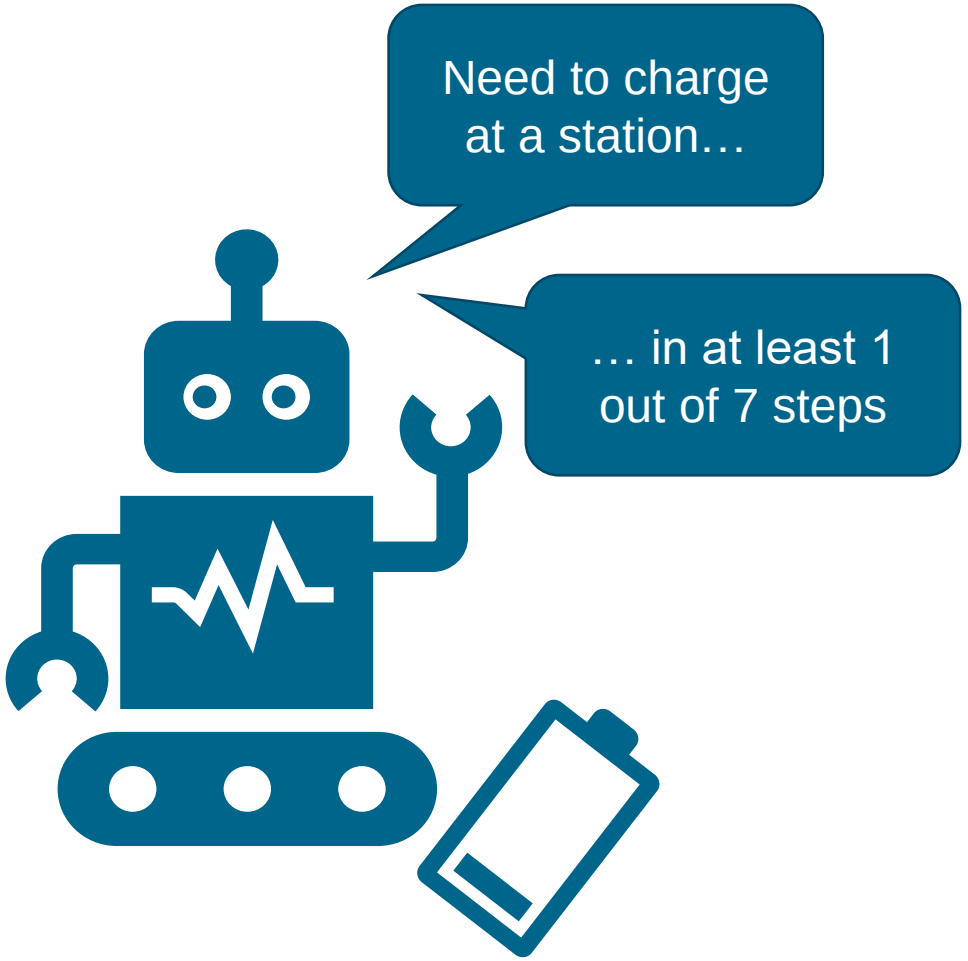


Window Counting Constraints



Window Counting Constraints

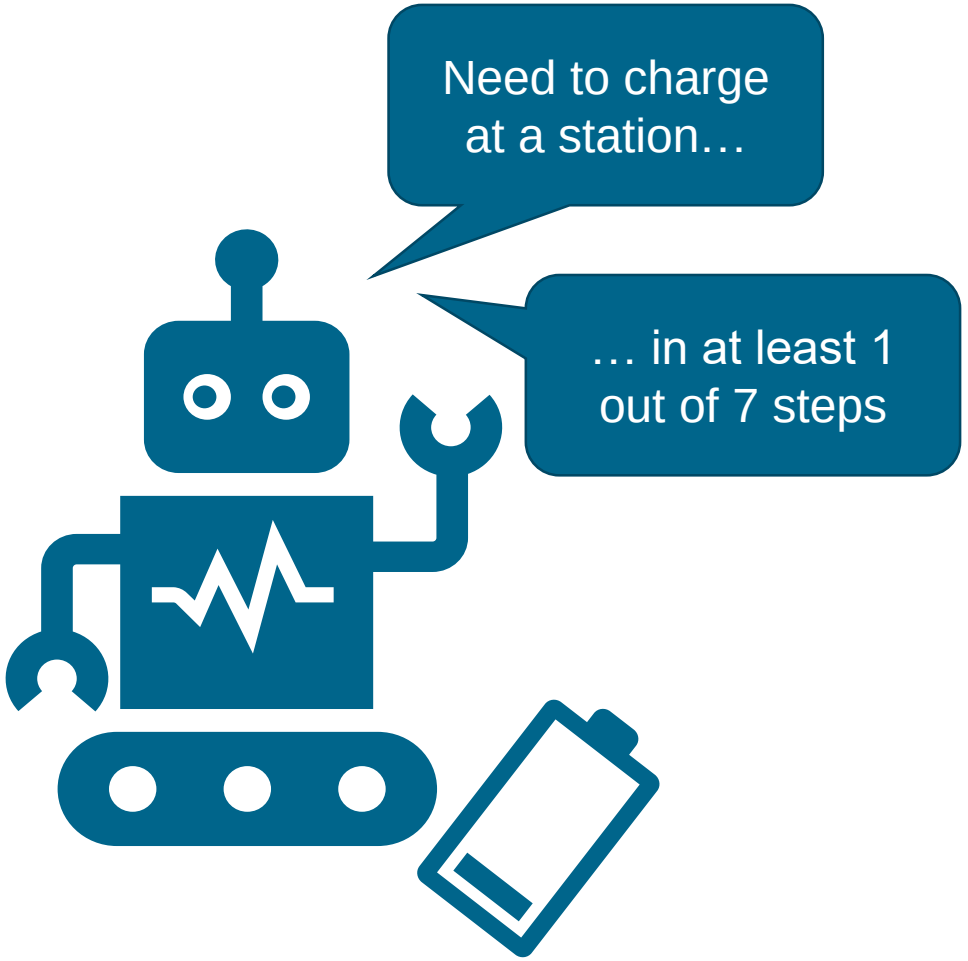
„The system plays an action c at least k times out of l turns.“



Window Counting Constraints

„The system plays an action c at least k times out of l turns.“

- *„at most“* instead of *„at least“* possible

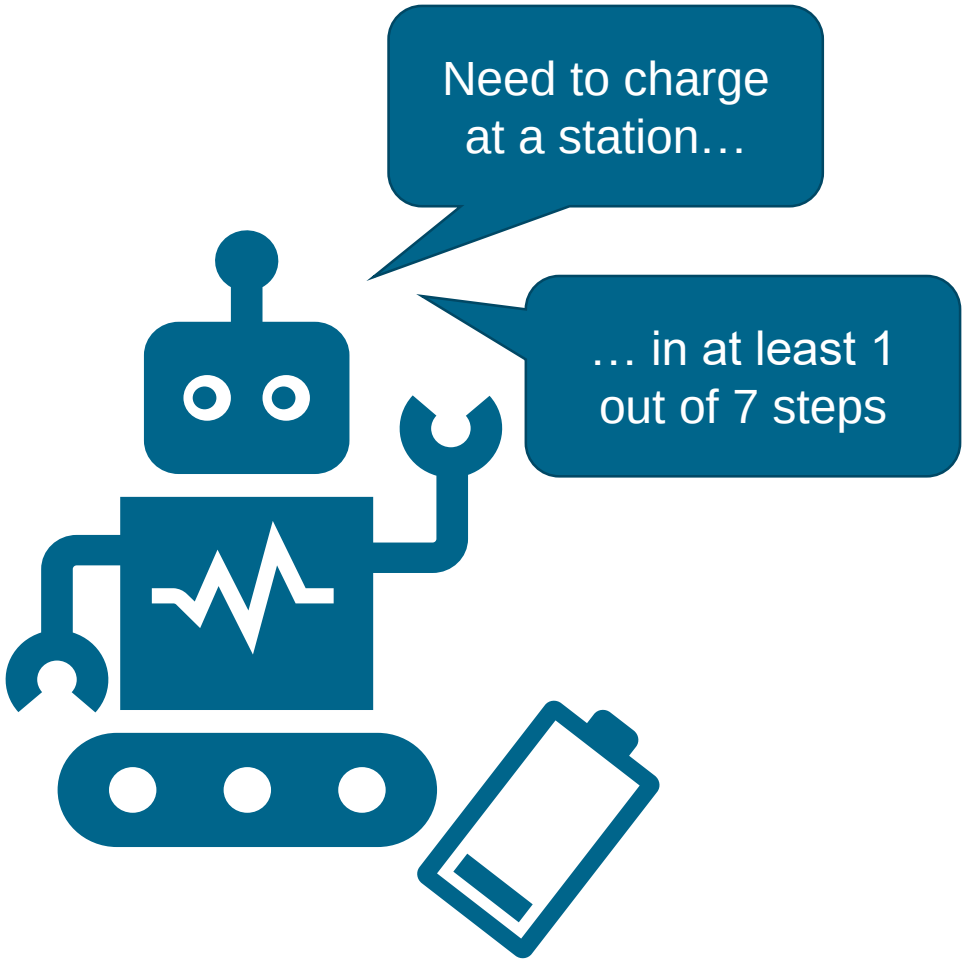


Window Counting Constraints

„The system plays an action c at least k times out of l turns.“

- *„at most“* instead of *„at least“* possible
- Constraint holds globally („sliding window“)
  …$bbcbbbcbbbbbbbcbbccccbbbc$…

  c at least 1 times out of 7 ✓



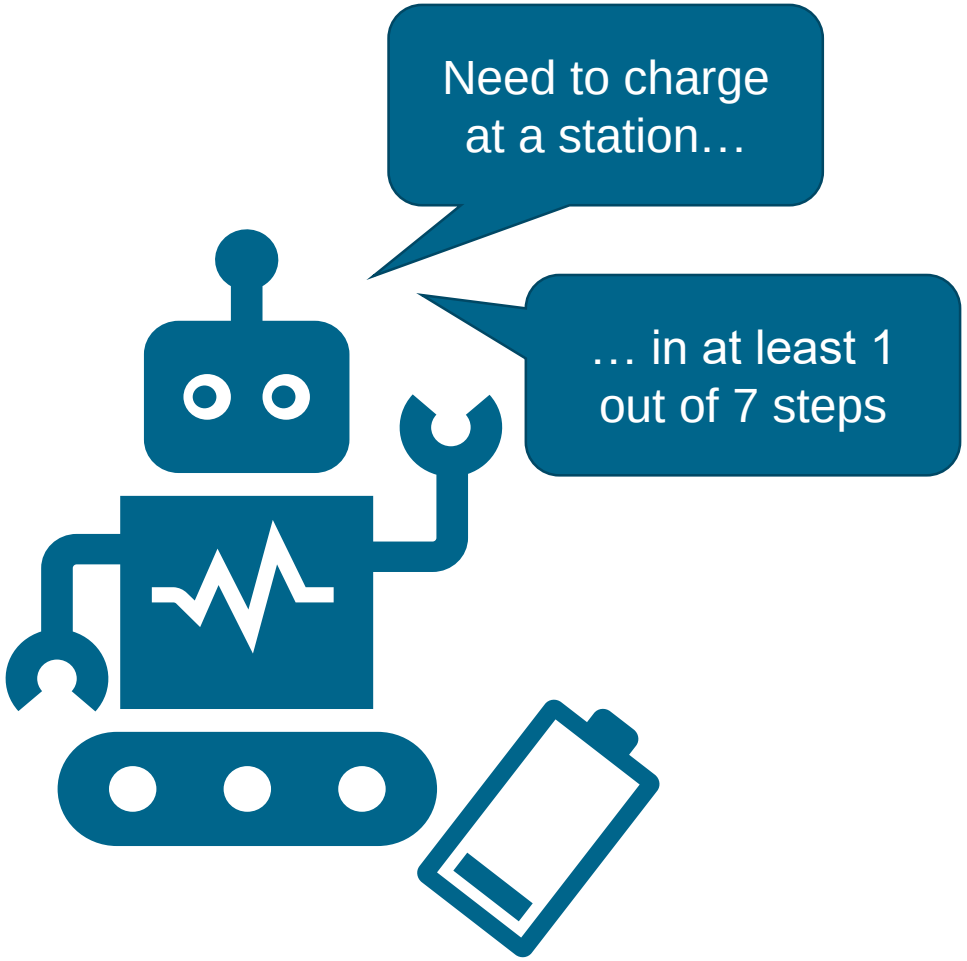
Window Counting Constraints

„The system plays an action c at least k times out of l turns.“

- *„at most“* instead of *„at least“* possible
- Constraint holds globally („sliding window“)

$\ldots bbcbbbcbbbbbbcbbccccbbbc\ldots$

c at least 1 times out of 7 ✓



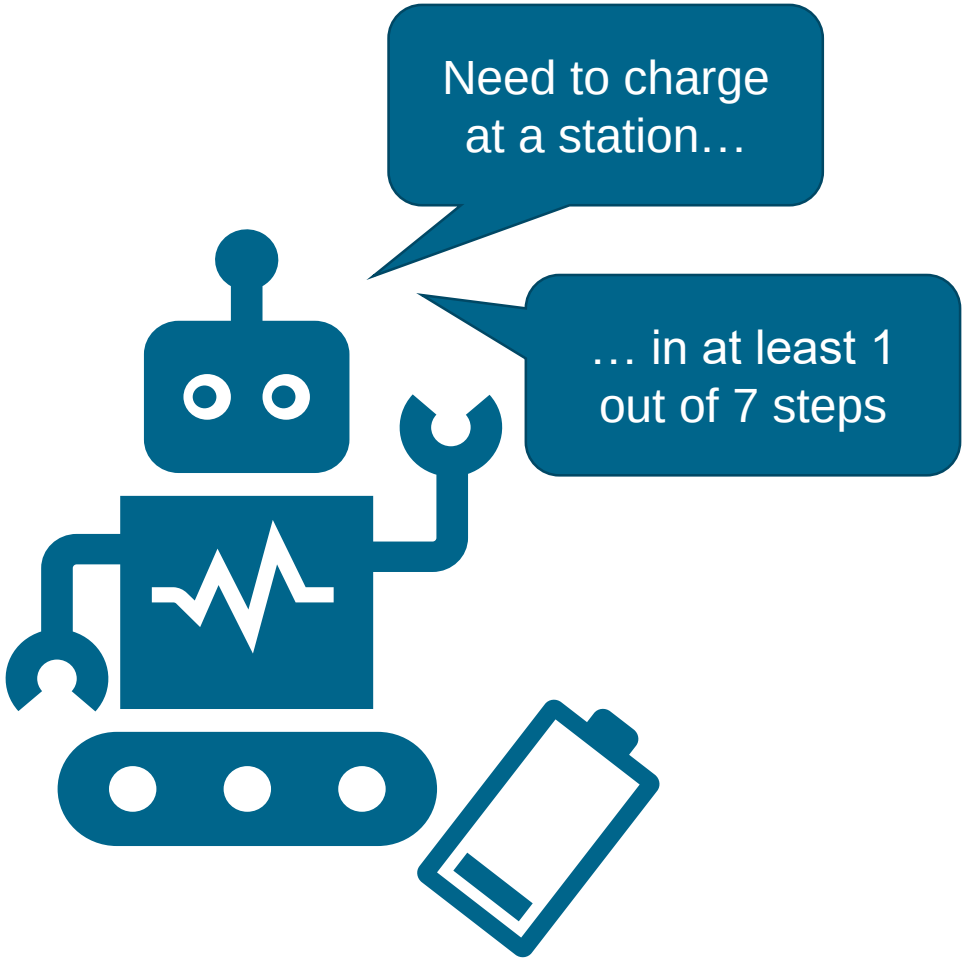
Window Counting Constraints

„The system plays an action c at least k times out of l turns.“

- *„at most“* instead of *„at least“* possible
- Constraint holds globally („sliding window“)

$\ldots bbcbbbcbbbbbbcbbccccbbbc\ldots$

c at least 1 times out of 7 ✓



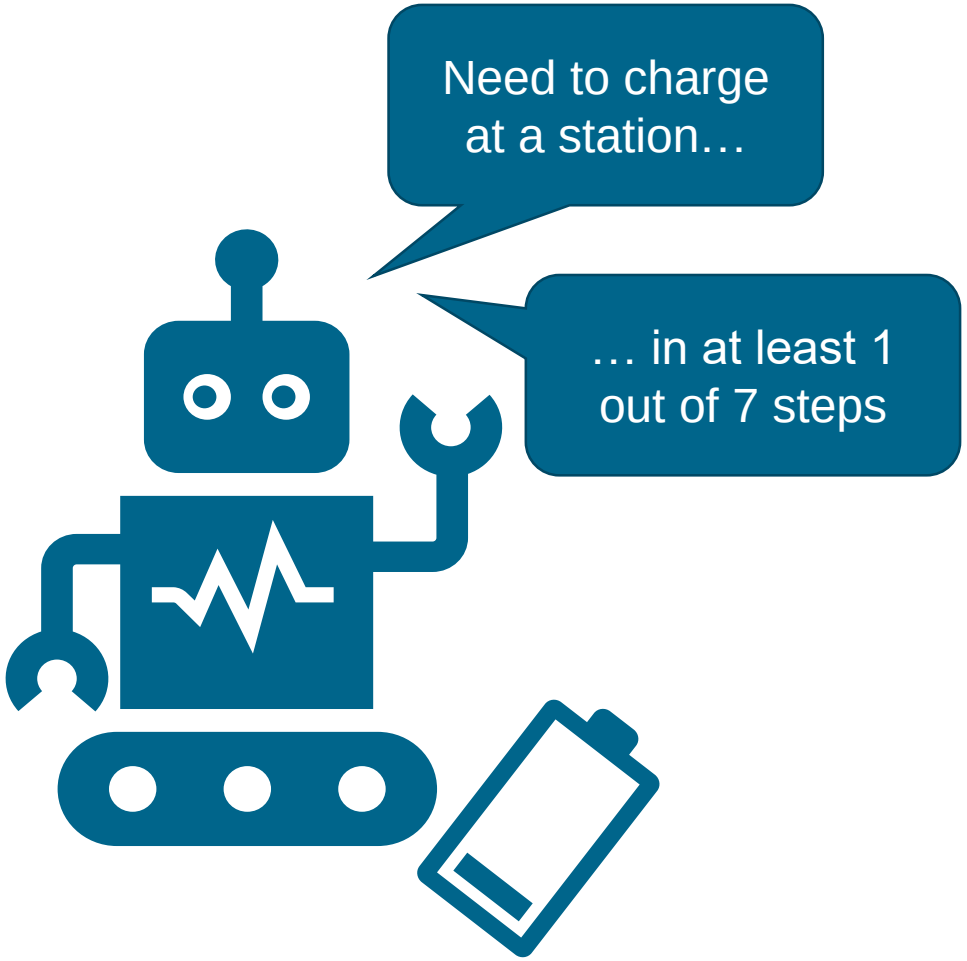
Window Counting Constraints

„The system plays an action c at least k times out of l turns.“

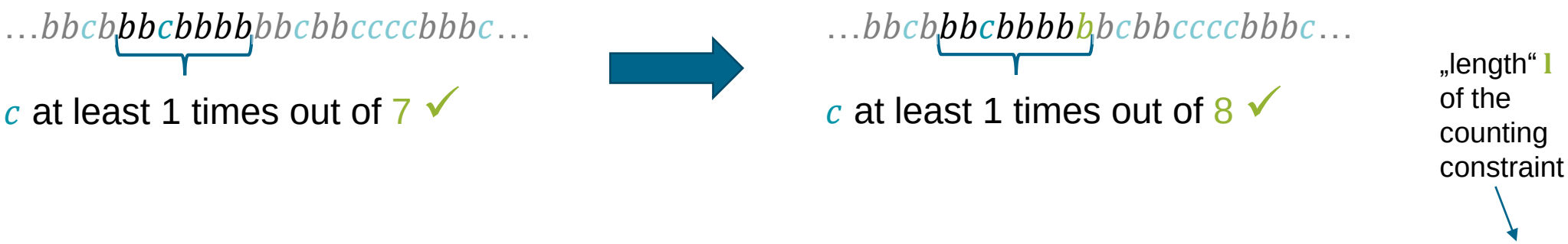
- *„at most“* instead of *„at least“* possible
- Constraint holds globally („sliding window“)

$\ldots bbcbbbcbbbbbbbcbbccccbbbc\ldots$

c at least 1 times out of 7 ✓

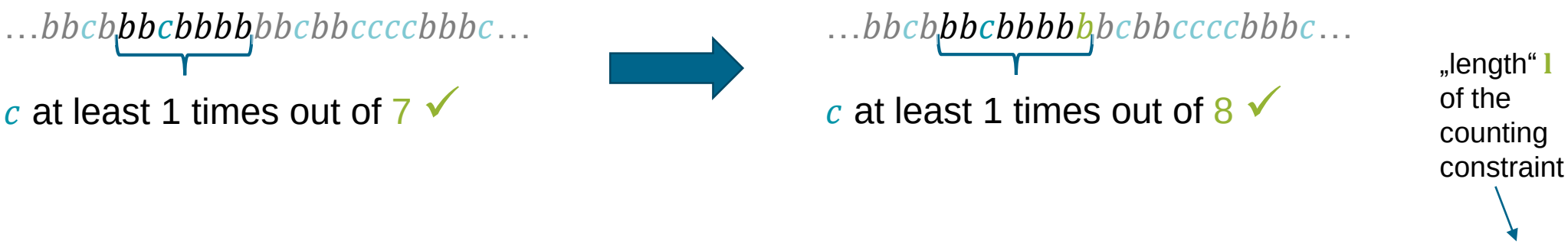


Synthesis with counting constraints in a nutshell



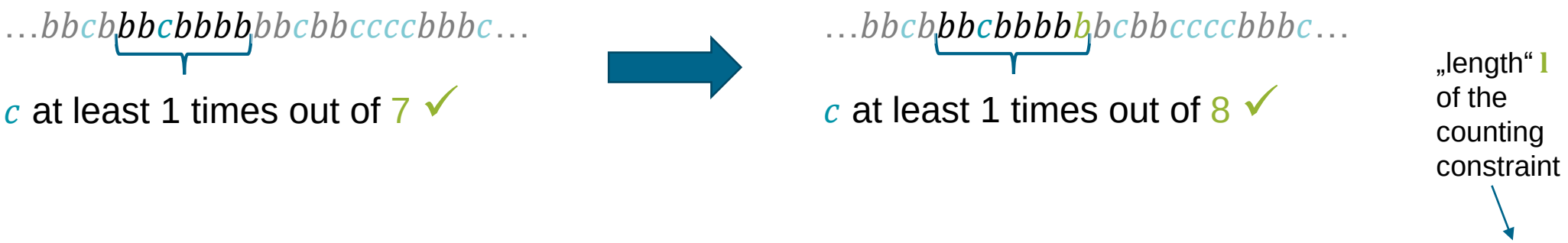
- A strategy that satisfies „The system plays an action c at least k times out of l turns.“ also satisfies „The system plays an action c at least k times out of m turns.“ for each $m > l$ (“monotone property”).

Synthesis with counting constraints in a nutshell



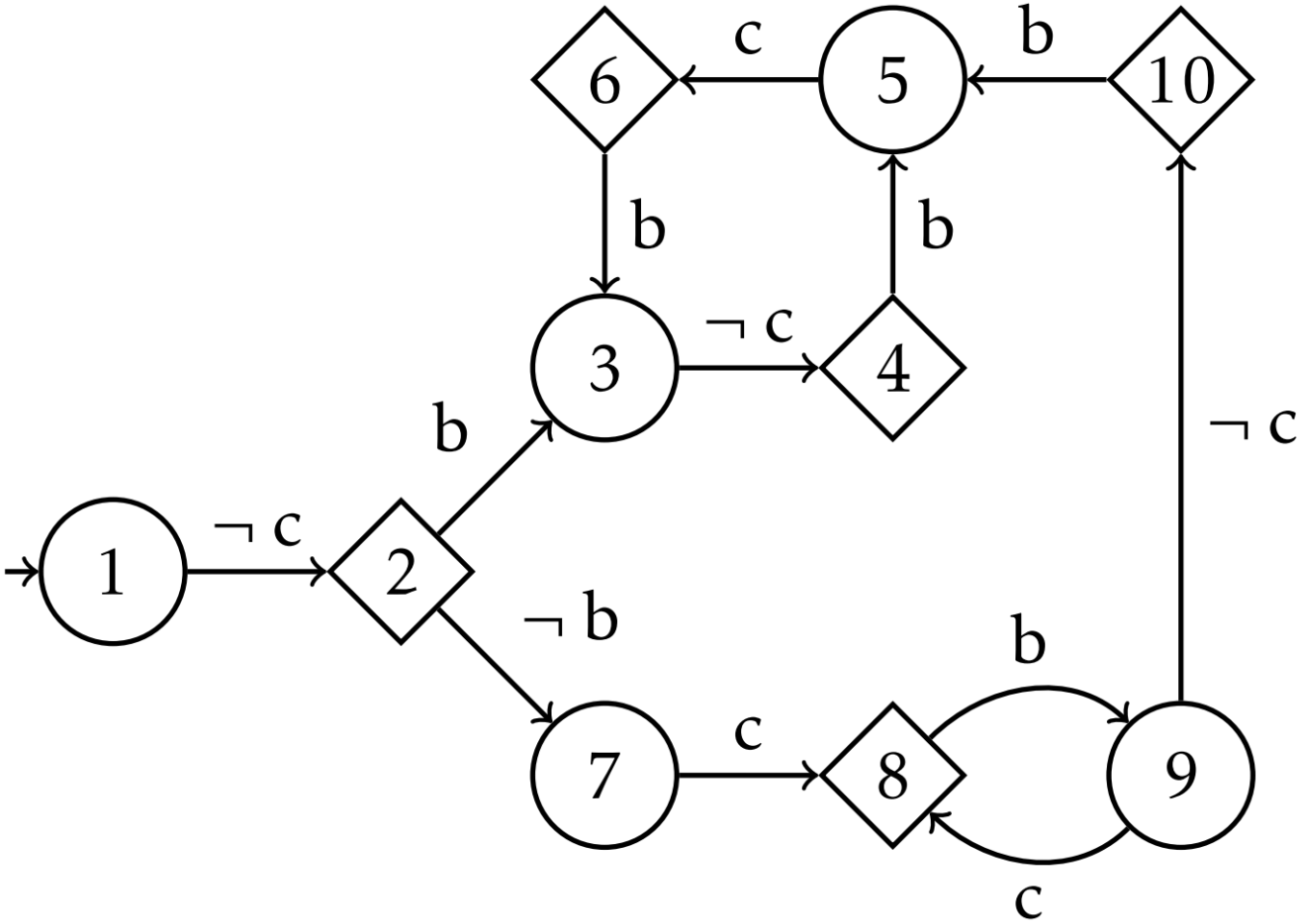
- A strategy that satisfies „The system plays an action c at least k times out of l turns.“ also satisfies „The system plays an action c at least k times out of m turns.“ for each $m > l$ (“monotone property”).
- Use information on winnable states for shorter constraint length for reducing size of required graphs!

Synthesis with counting constraints in a nutshell



- A strategy that satisfies „The system plays an action c at least k times out of l turns.“ also satisfies „The system plays an action c at least k times out of m turns.“ for each $m > l$ (“monotone property”).
- Use information on winnable states for shorter constraint length for reducing size of required graphs!
- Synthesis with iteration over counting constraint length

Synthesis with Counting Constraints - Example



EGO plays c at least 1 time in 7 turns.

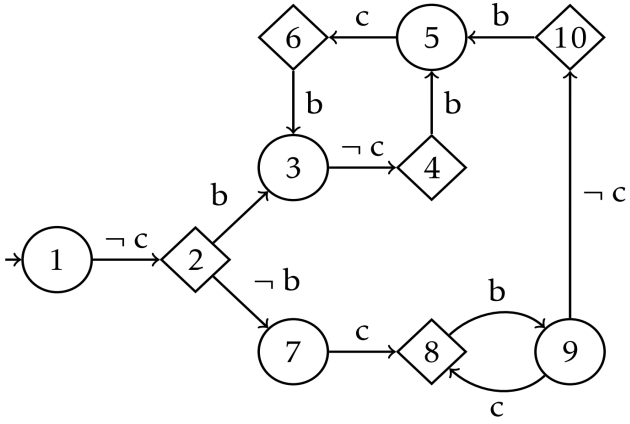
Synthesis with Counting Constraints - Example

Example

Algorithm

In: - EGO plays c at least k time in l turns.
- finite two-player game graph

Intermediate Results (Example)



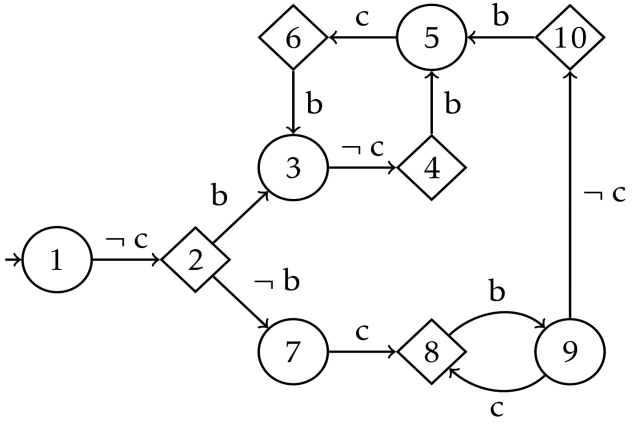
EGO plays c at least 1 time in 7 turns.

Synthesis with Counting Constraints - Example

Example

EGO plays c at least 1 time in 1 turn.

Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

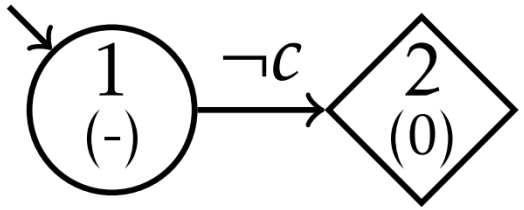
Algorithm

In: - EGO plays c at least k time in l turns.
- finite two-player game graph

$$l' \leftarrow k$$

Synthesis with Counting Constraints - Example

Example



EGO plays c at least 1 time in 1 turn.

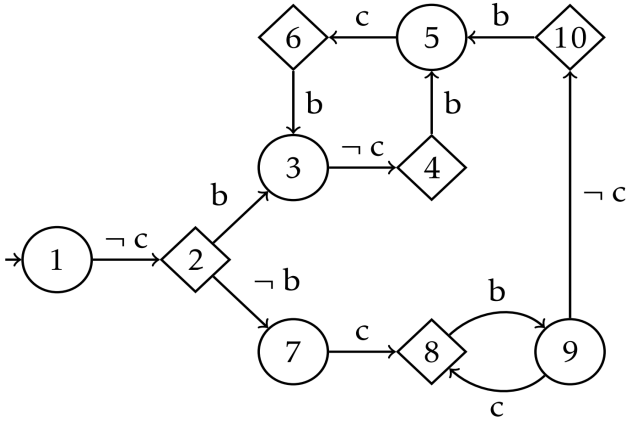
Algorithm

In: - EGO plays c at least k time in l turns.
- finite two-player game graph

$l' \leftarrow k$

(partly) construct game graph with memory l'

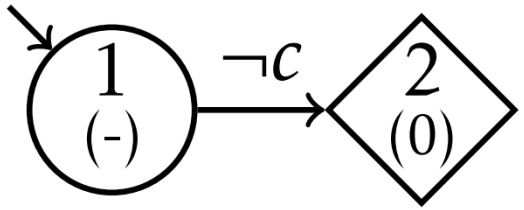
Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

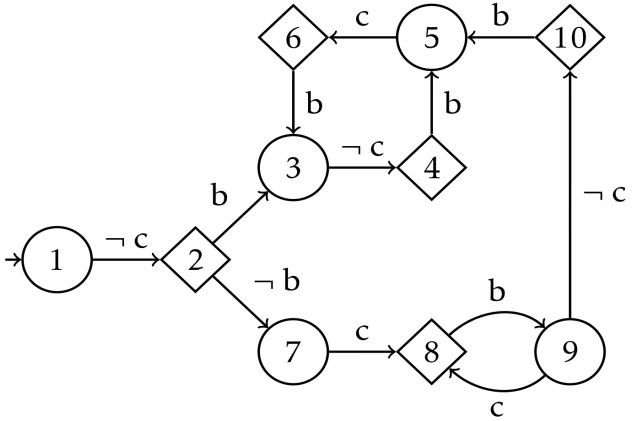
Synthesis with Counting Constraints - Example

Example



EGO plays c at least 1 time in 1 turn.

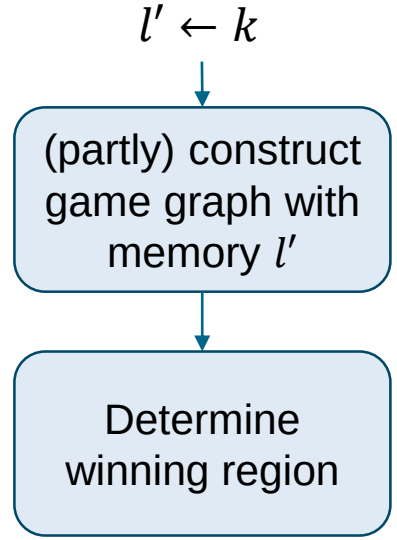
Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

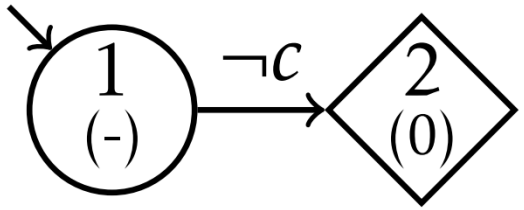
Algorithm

In: - EGO plays c at least k time in l turns.
- finite two-player game graph



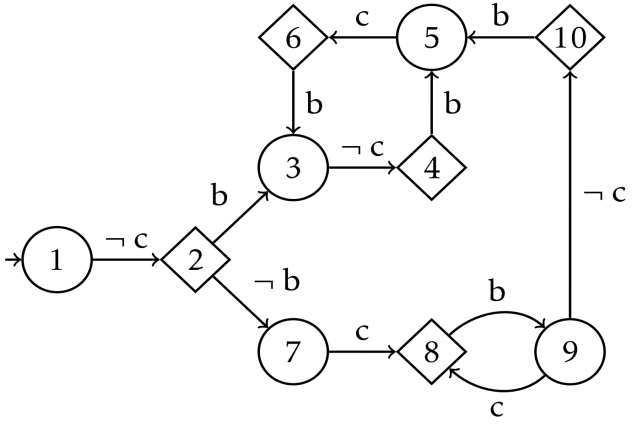
Synthesis with Counting Constraints - Example

Example



EGO plays c at least 1 time in 1 turn.

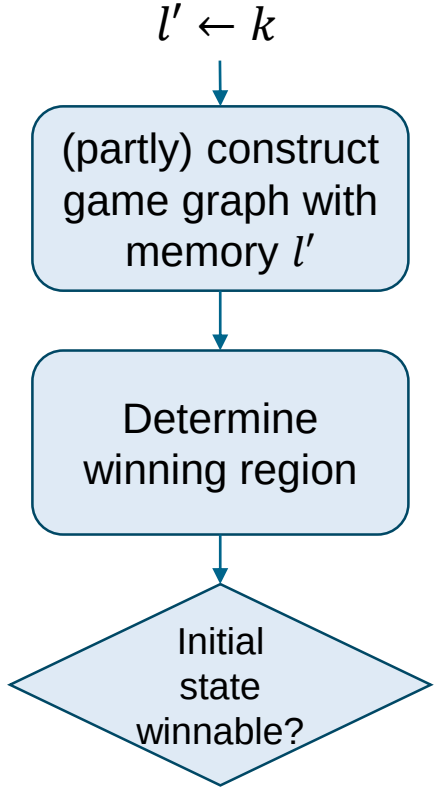
Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

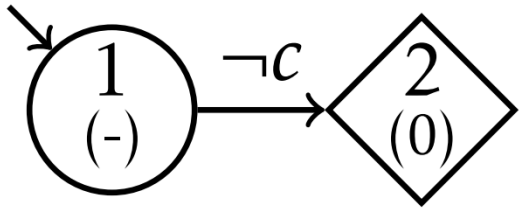
Algorithm

In: - EGO plays c at least k time in l turns.
- finite two-player game graph



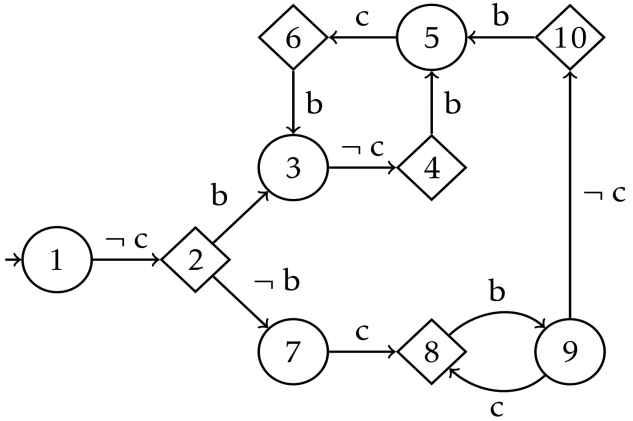
Synthesis with Counting Constraints - Example

Example



EGO plays c at least 1 time in 1 turn.

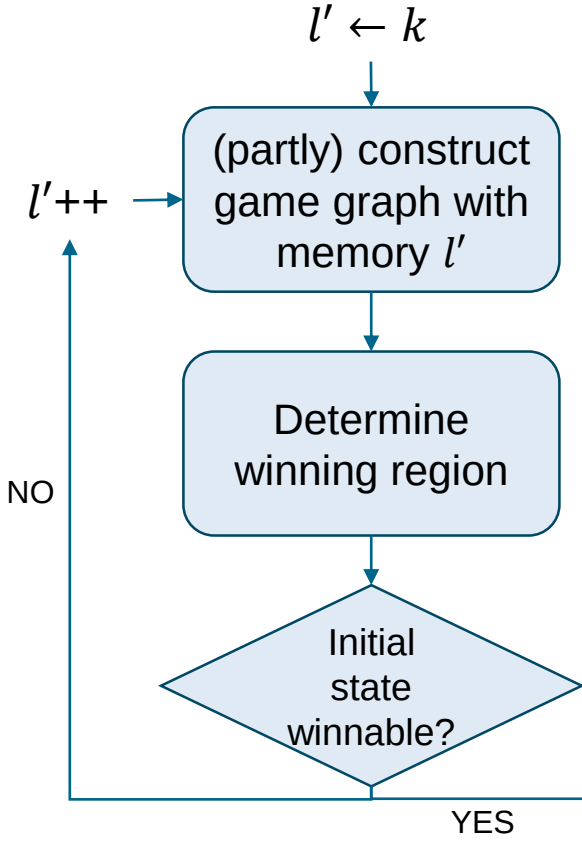
Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

Algorithm

In: - EGO plays c at least k time in l turns.
- finite two-player game graph

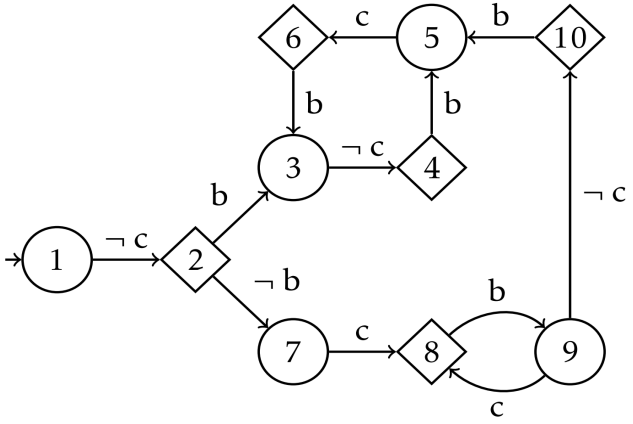


Synthesis with Counting Constraints - Example

Example

EGO plays c at least 1 time in 2 turns.

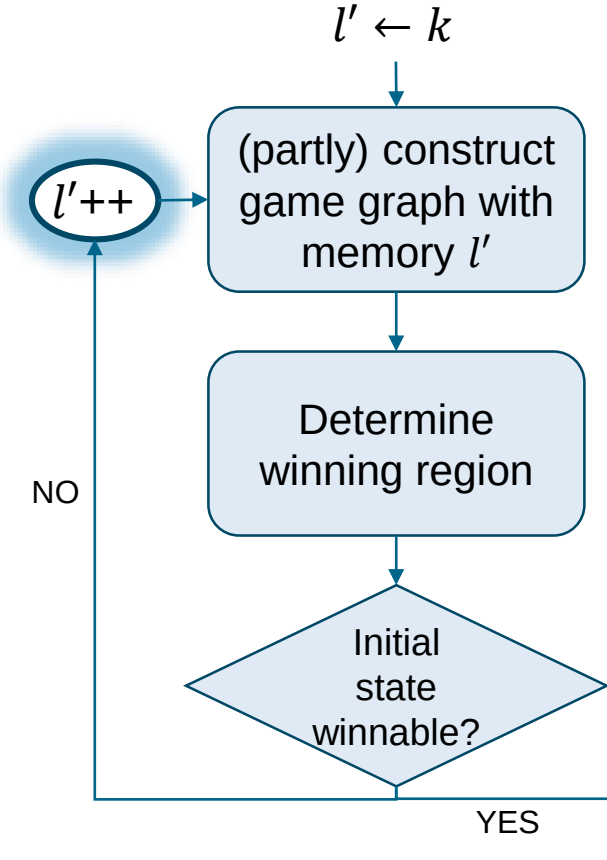
Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

Algorithm

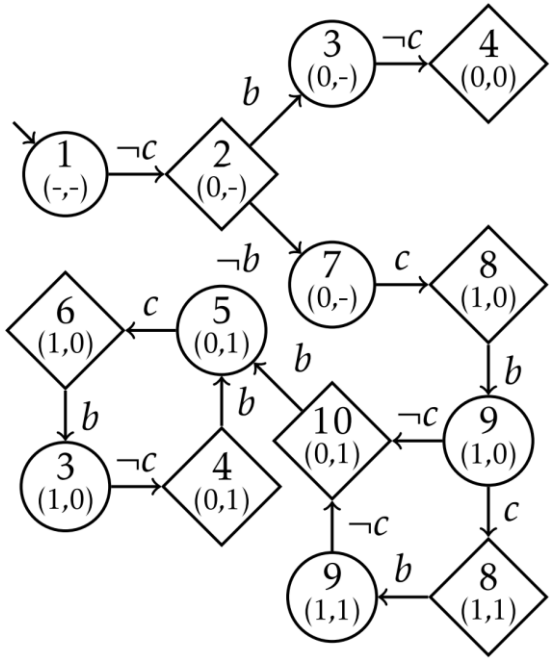
In: - EGO plays c at least k time in l turns.
- finite two-player game graph



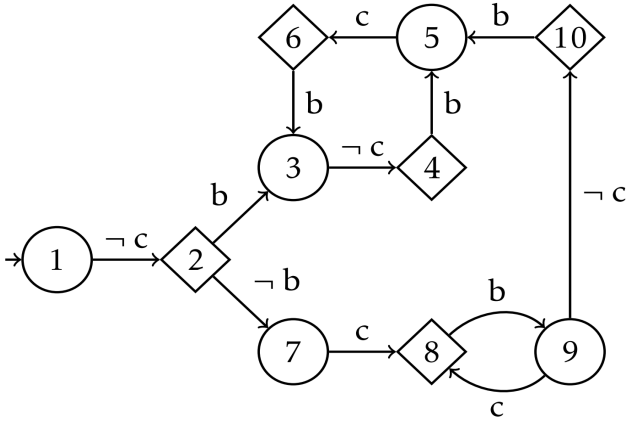
Synthesis with Counting Constraints - Example

Example

EGO plays c at least 1 time in 2 turns.



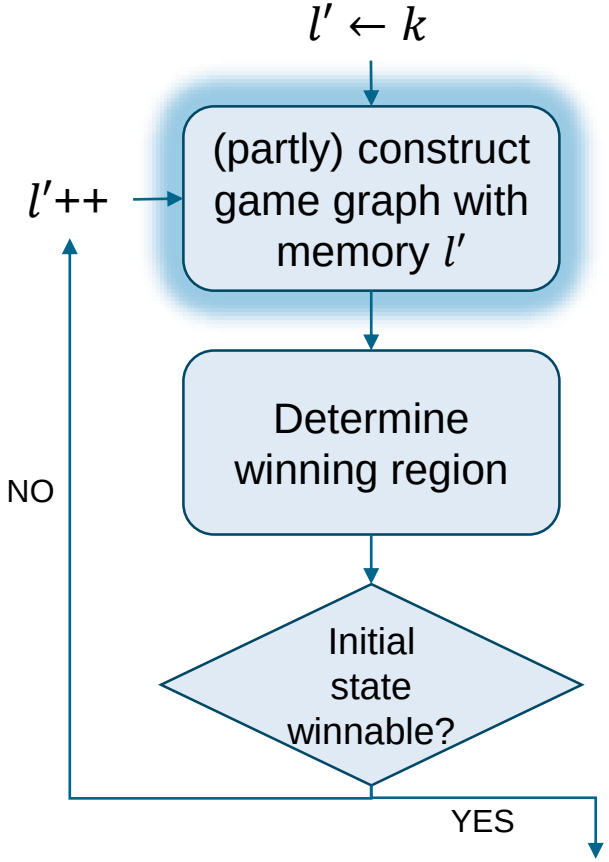
Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

Algorithm

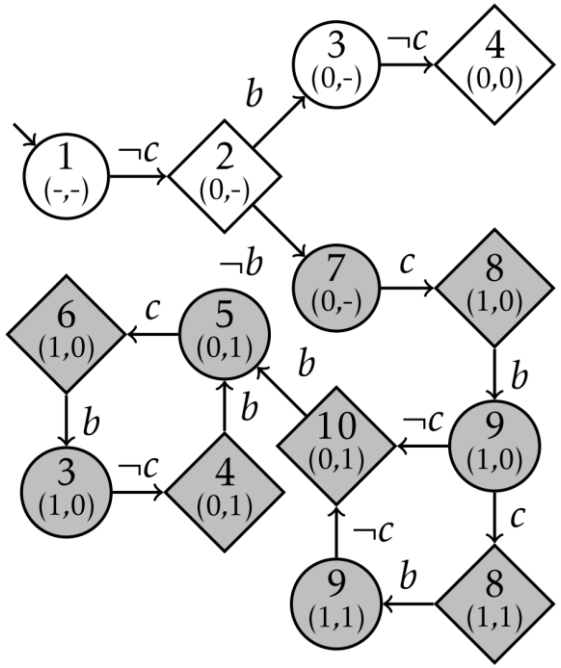
In: - EGO plays c at least k time in l turns.
- finite two-player game graph



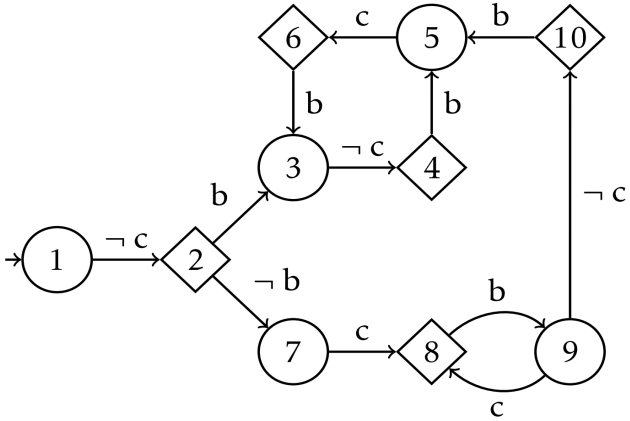
Synthesis with Counting Constraints - Example

Example

EGO plays c at least 1 time in 2 turns.



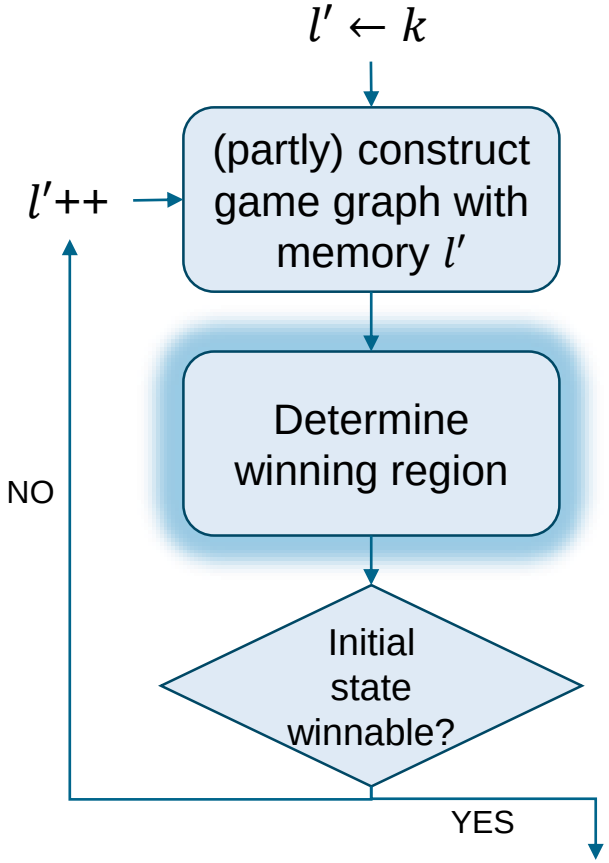
Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

Algorithm

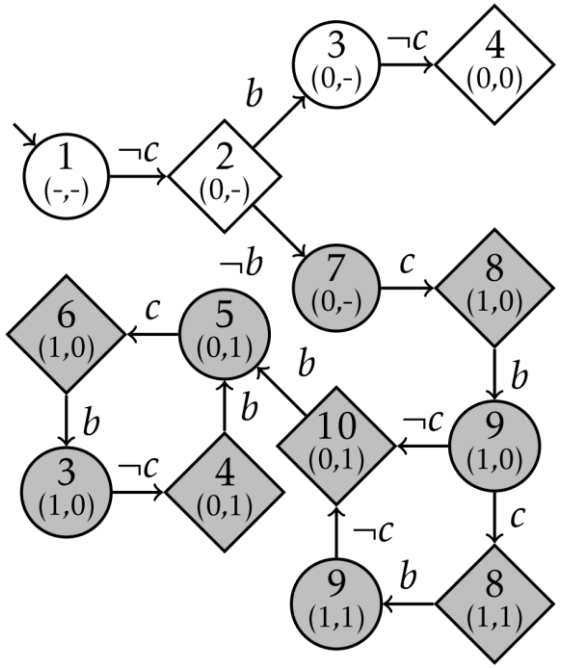
In: - EGO plays c at least k time in l turns.
- finite two-player game graph



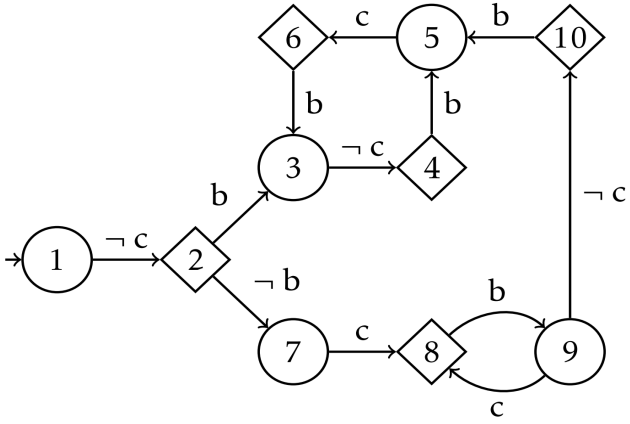
Synthesis with Counting Constraints - Example

Example

EGO plays c at least 1 time in 2 turns.



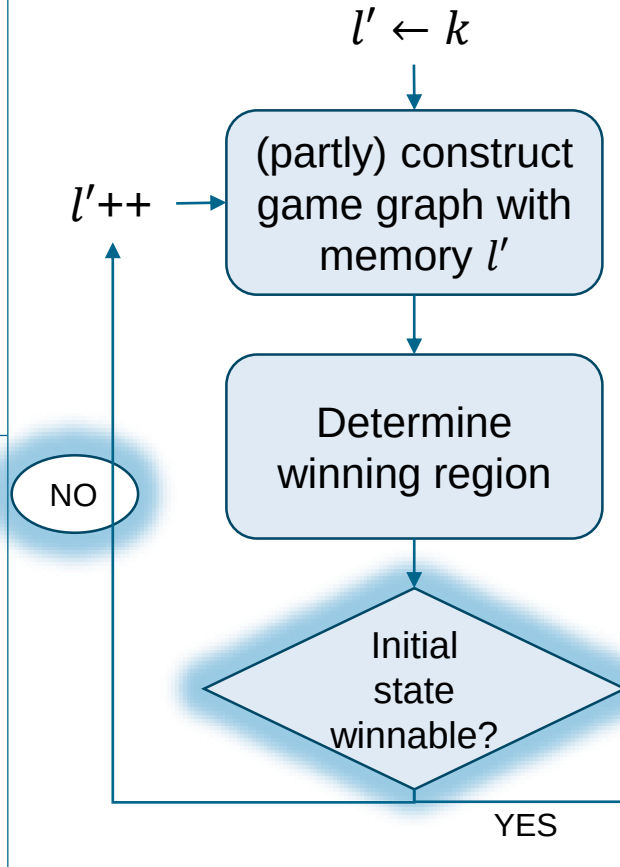
Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

Algorithm

In: - EGO plays c at least k time in l turns.
- finite two-player game graph

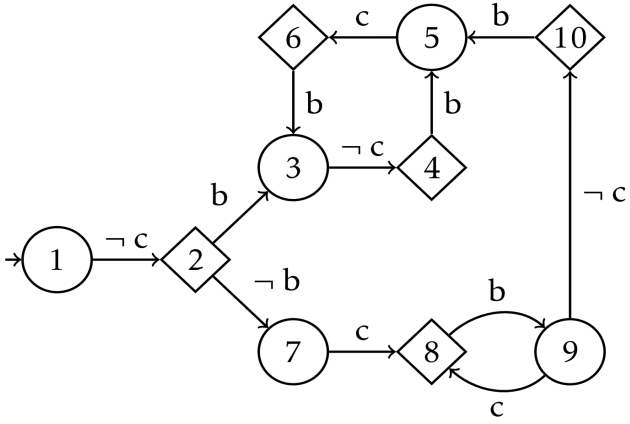


Synthesis with Counting Constraints - Example

Example

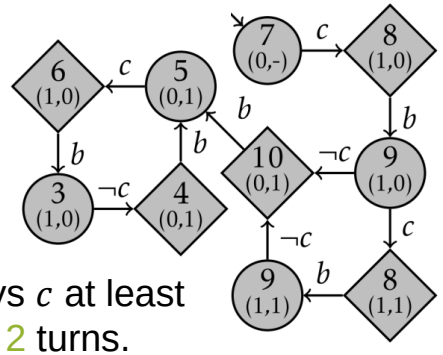
EGO plays c at least 1 time in 3 turns.

Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

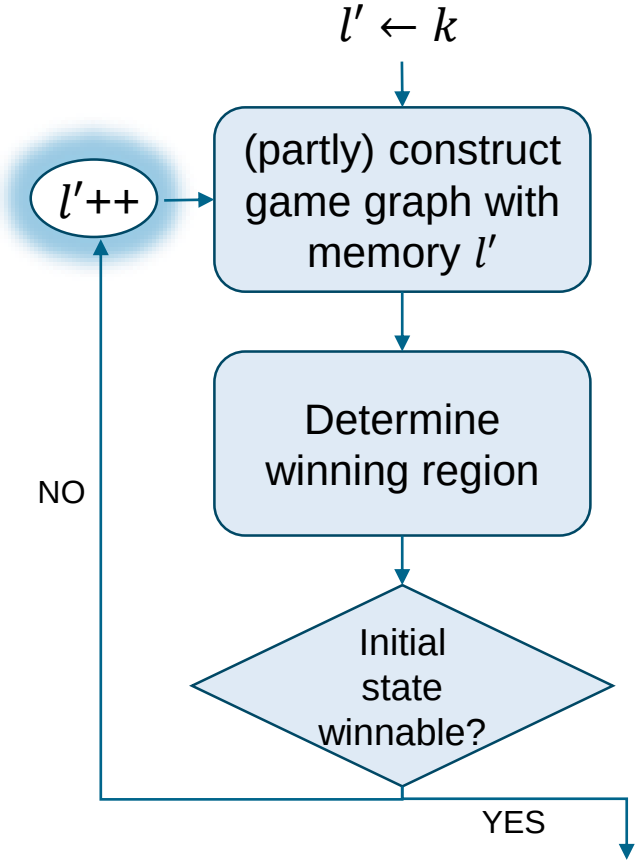
Winning region of iteration 2



EGO plays c at least 1 time in 2 turns.

Algorithm

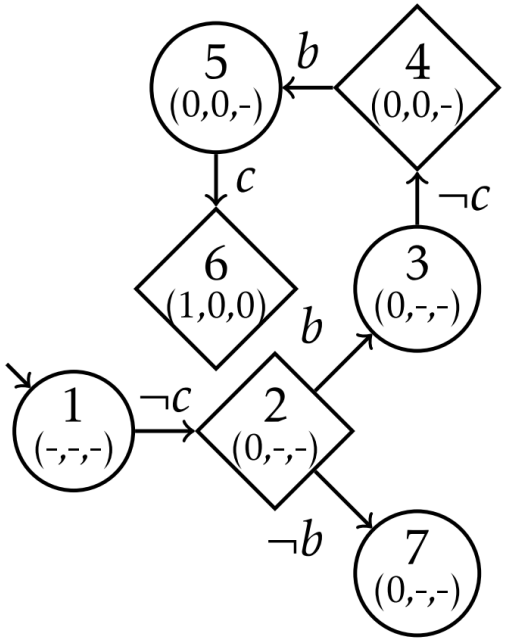
In: - EGO plays c at least k time in l turns.
- finite two-player game graph



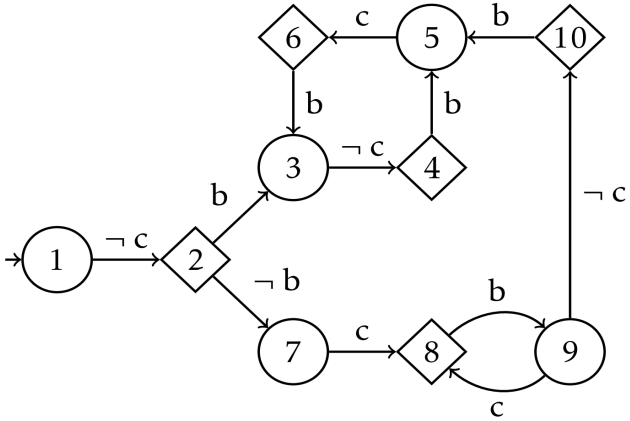
Synthesis with Counting Constraints - Example

Example

EGO plays c at least 1 time in 3 turns.

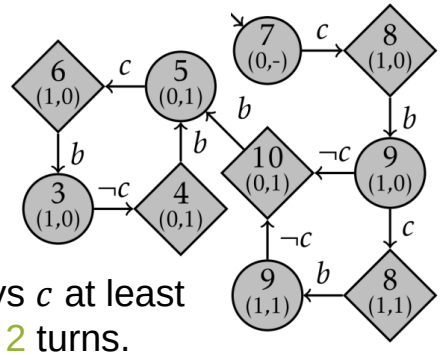


Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

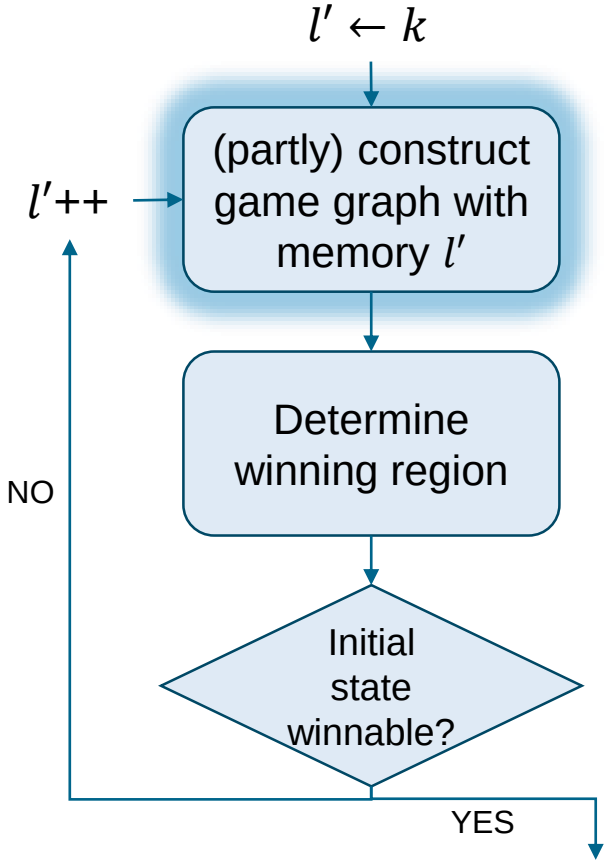
Winning region of iteration 2



EGO plays c at least 1 time in 2 turns.

Algorithm

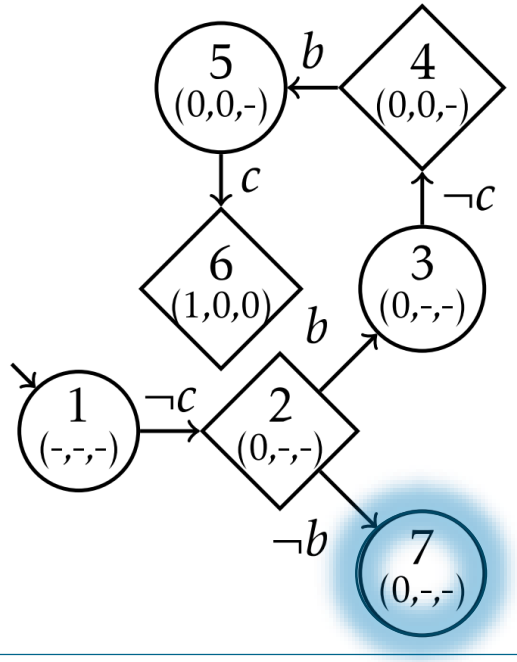
In: - EGO plays c at least k time in l turns.
- finite two-player game graph



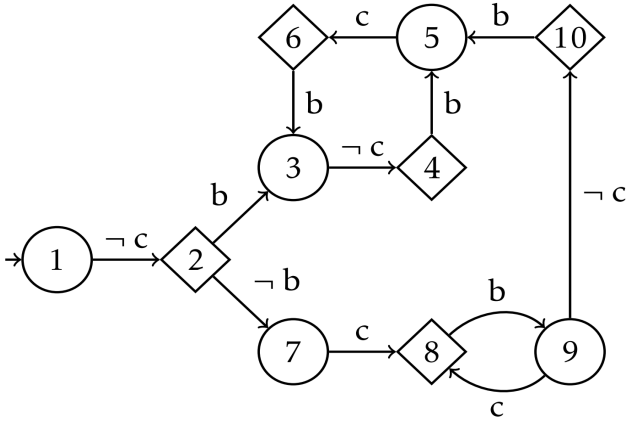
Synthesis with Counting Constraints - Example

Example

EGO plays c at least 1 time in 3 turns.

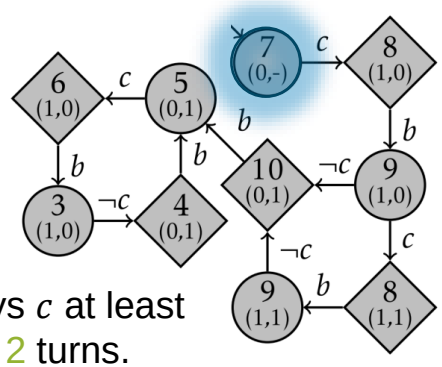


Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

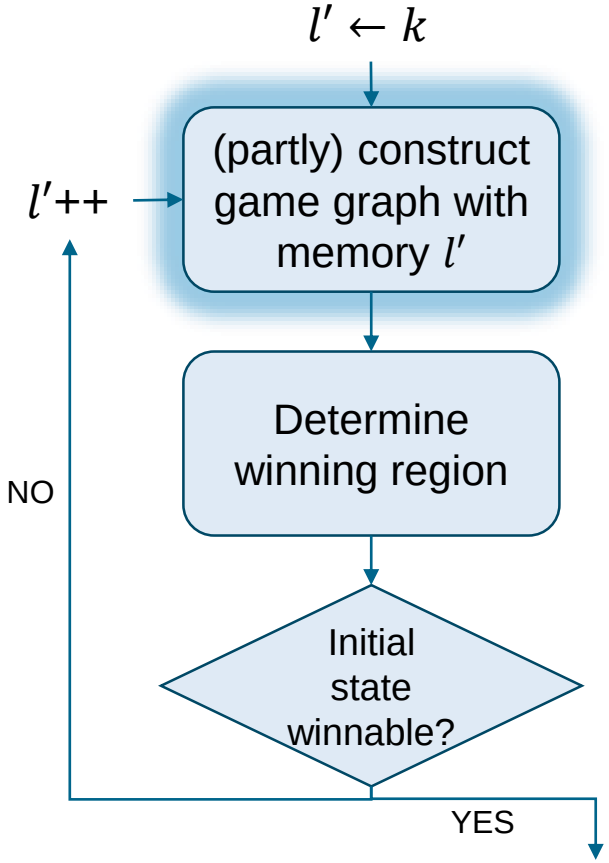
Winning region of iteration 2



EGO plays c at least 1 time in 2 turns.

Algorithm

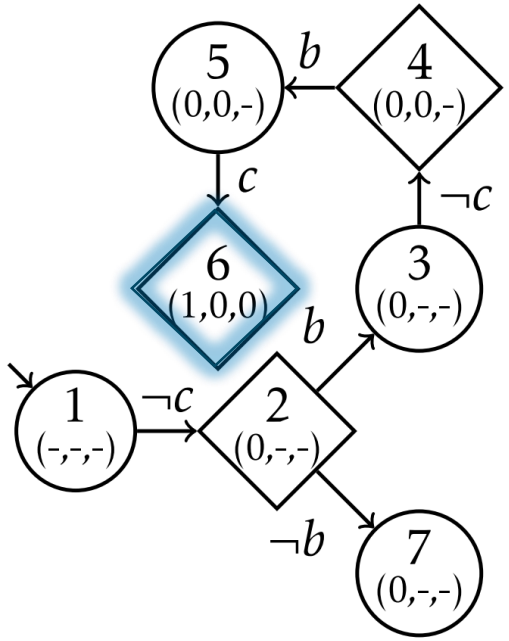
In: - EGO plays c at least k time in l turns.
- finite two-player game graph



Synthesis with Counting Constraints - Example

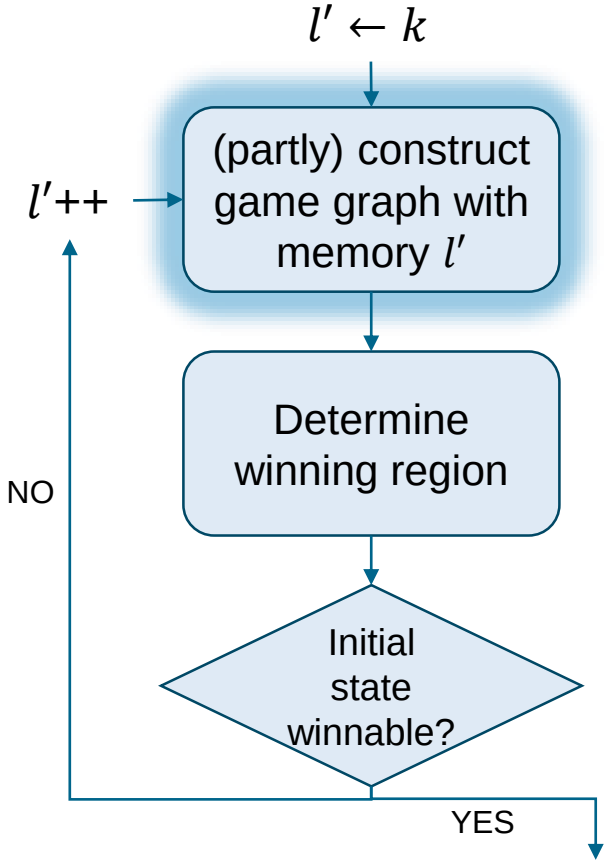
Example

EGO plays c at least 1 time in 3 turns.

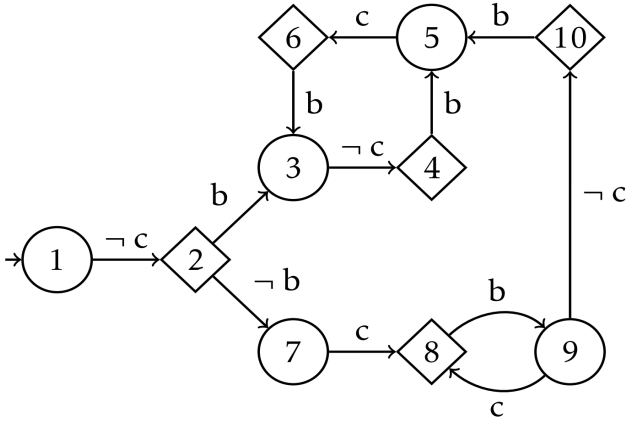


Algorithm

In: - EGO plays c at least k time in l turns.
- finite two-player game graph

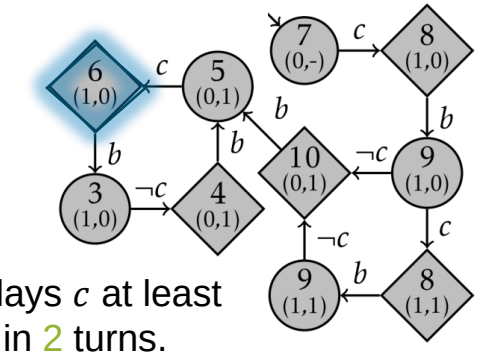


Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

Winning region of iteration 2

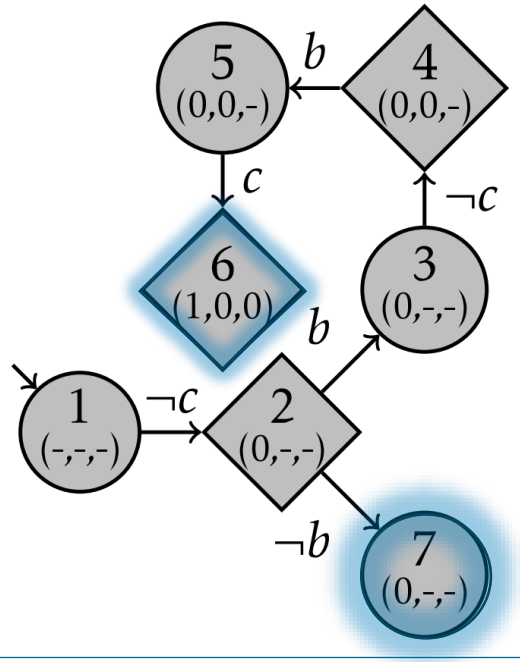


EGO plays c at least 1 time in 2 turns.

Synthesis with Counting Constraints - Example

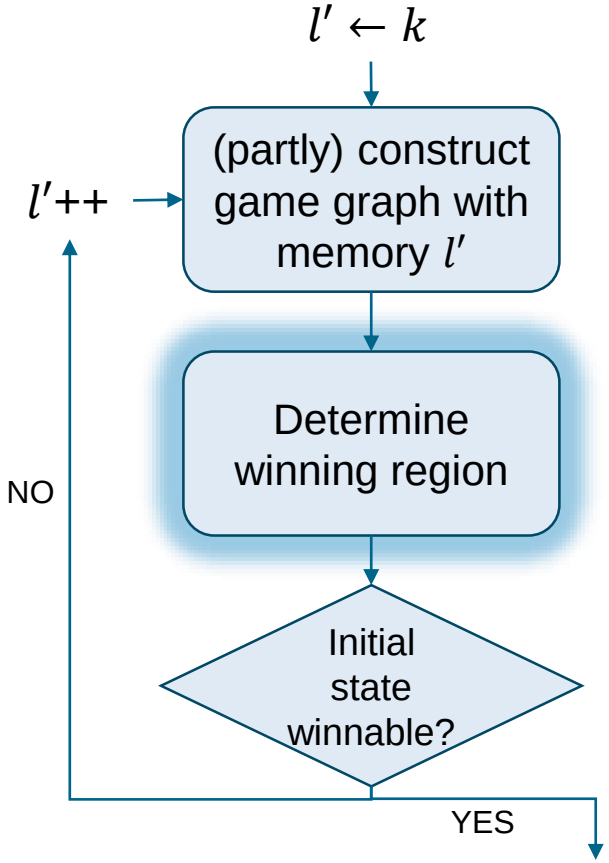
Example

EGO plays c at least 1 time in 3 turns.

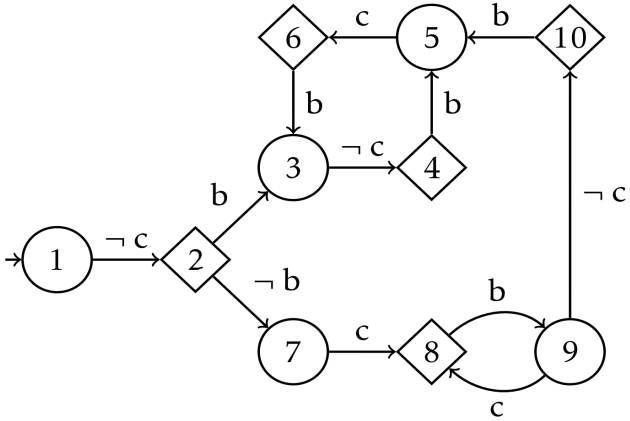


Algorithm

In: - EGO plays c at least k time in l turns.
- finite two-player game graph

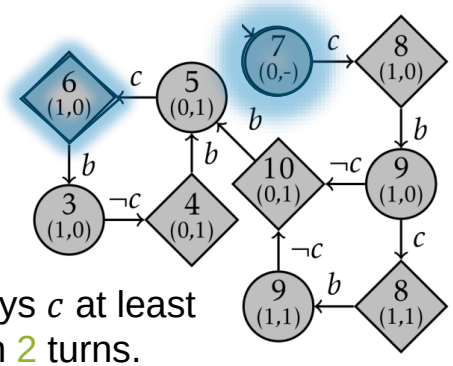


Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

Winning region of iteration 2

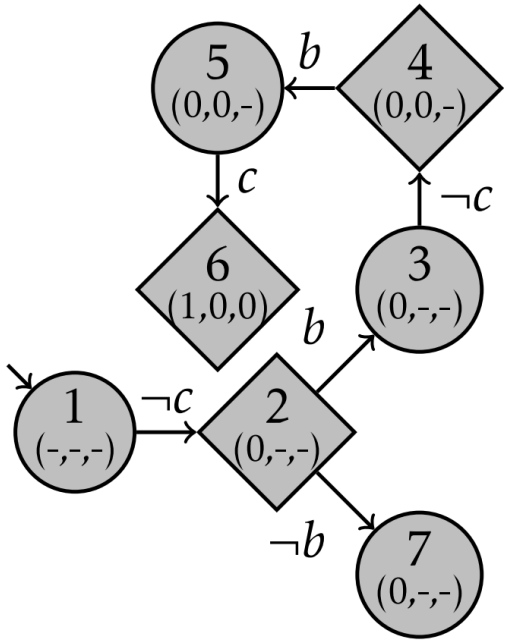


EGO plays c at least 1 time in 2 turns.

Synthesis with Counting Constraints - Example

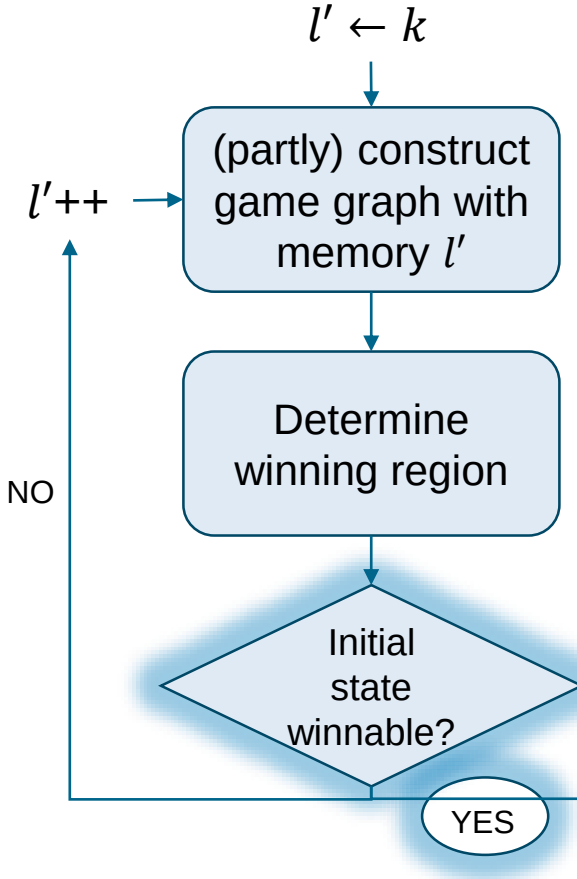
Example

EGO plays c at least 1 time in 3 turns.

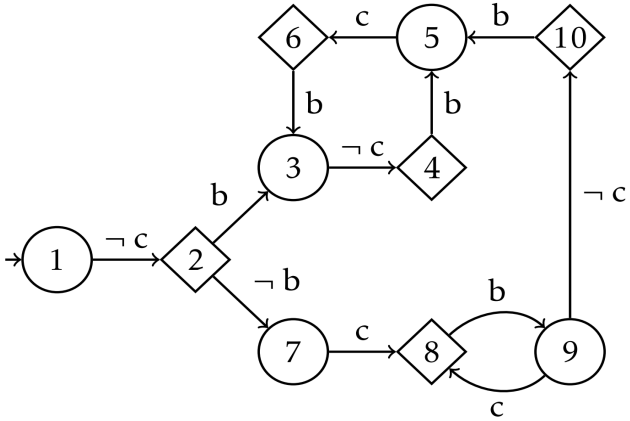


Algorithm

In: - EGO plays c at least k time in l turns.
- finite two-player game graph

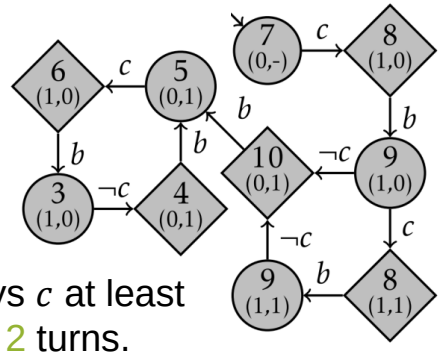


Intermediate Results (Example)



EGO plays c at least 1 time in 7 turns.

Winning region of iteration 2



EGO plays c at least 1 time in 2 turns.

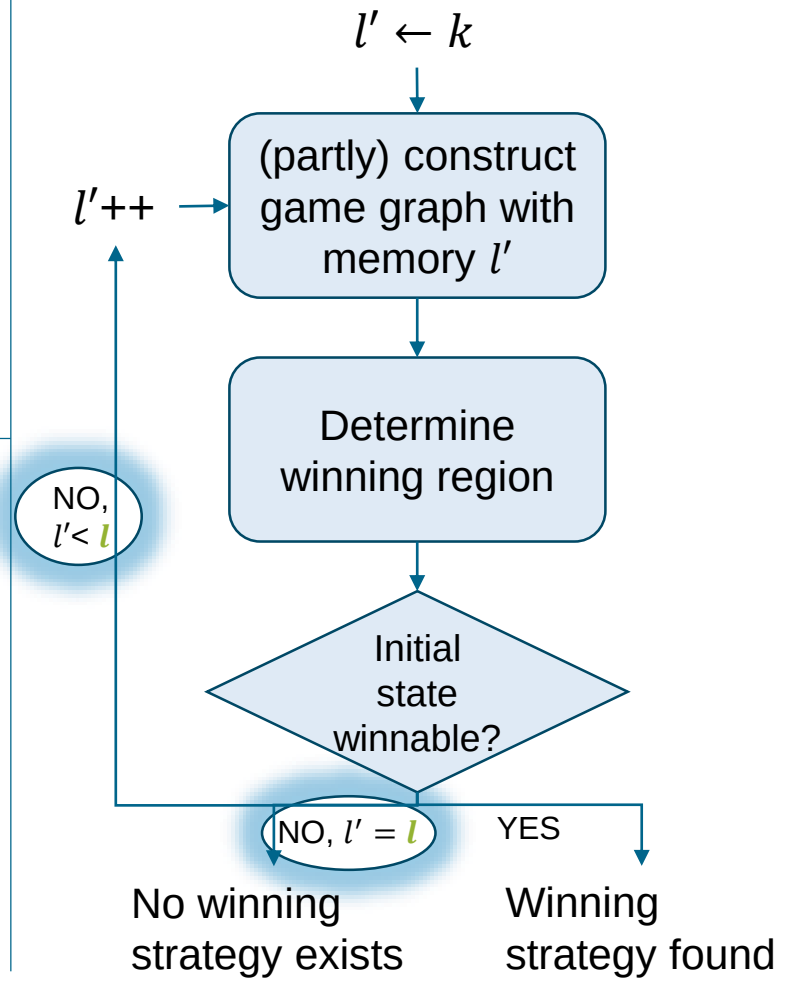
Synthesis with Counting Constraints - Example

Example

Intermediate Results (Example)

Algorithm

In: - EGO plays c at least k time in l turns.
- finite two-player game graph



Experimental Results

Setting:

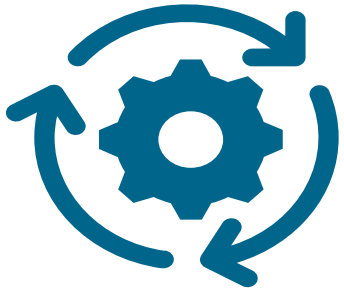
- Counting constraint „The system plays c at least 3 times in 10 turns.“
- Game graph with 1.8M states & 2.7M transitions

Experimental Results

Setting:

- Counting constraint „The system plays c at least 3 times in 10 turns.“
- Game graph with 1.8M states & 2.7M transitions

Implementation in Python

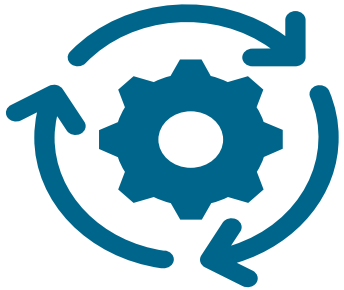


Experimental Results

Setting:

- Counting constraint „The system plays c at least 3 times in 10 turns.“
- Game graph with 1.8M states & 2.7M transitions

Implementation in Python



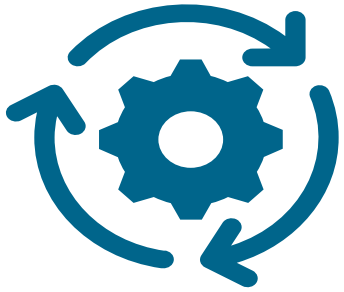
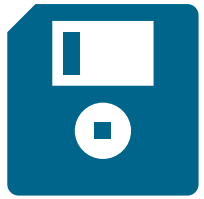
Reduction of time for synthesis

4 times faster

Experimental Results

Setting:

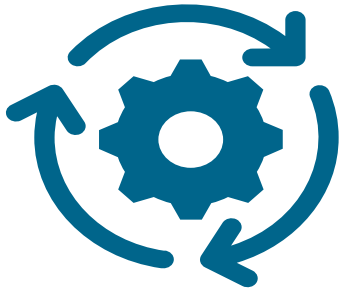
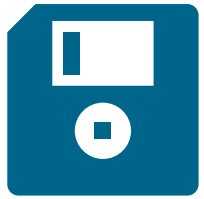
- Counting constraint „The system plays c at least 3 times in 10 turns.“
- Game graph with 1.8M states & 2.7M transitions

Implementation in Python	Reduction of time for synthesis	Reduction of number of states
	 4 times faster	 60% less states

Experimental Results

Setting:

- Counting constraint „The system plays c at least 3 times in 10 turns.“
- Game graph with 1.8M states & 2.7M transitions

Implementation in Python	Reduction of time for synthesis	Reduction of number of states
	 4 times faster	 60% less states

- Synthesis with successively enlarged couting constraints & incomplete graph construction shows great potential
- Proof of concept successful

Future Work

Towards cooperative games

- Consider counting constraints on behaviour of the second player
- Level of cooperation and synchonisation between players to be determined

Future Work

Towards cooperative games

- Consider counting constraints on behaviour of the second player
- Level of cooperation and synchonisation between players to be determined

Extension of counting constraint types

- Iterative approach for more specification types
- e.g. „If x is played, the system plays y after at most k turns“

Future Work

Towards cooperative games

- Consider counting constraints on behaviour of the second player
- Level of cooperation and synchonisation between players to be determined

Extension of counting constraint types

- Iterative approach for more specification types
- e.g. „If x is played, the system plays y after at most k turns“

Combination of various counting constraints

- Iterate over several counting constraints
- Successively or alternating

Future Work

Towards cooperative games

- Consider counting constraints on behaviour of the second player
- Level of cooperation and synchonisation between players to be determined

Extension of counting constraint types

- Iterative approach for more specification types
- e.g. „If x is played, the system plays y after at most k turns“

Combination of various counting constraints

- Iterate over several counting constraints
- Successively or alternating

Symbolic representation

- Avoid explicit representation of states in game graph
- Comparision of explicit and symbolic synthesis

Conclusion

Challenge

- Efficiency of synthesis algorithms for reactive systems

Conclusion

Challenge

- Efficiency of synthesis algorithms for reactive systems

Approach

- Exploitation of specific properties in the specification, namely: monotony in counting constraints
- Successive increase of constraint length
- Usage of retrieved knowledge of previous steps allows for incomplete automata construction

Conclusion

Challenge

- Efficiency of synthesis algorithms for reactive systems

Approach

- Exploitation of specific properties in the specification, namely: monotony in counting constraints
- Successive increase of constraint length
- Usage of retrieved knowledge of previous steps allows for incomplete automata construction

Results

- Algorithm for synthesis with counting constraints
- Experimental results show promising savings in computational time & memory
- Future reseach directions determined

Conclusion

Challenge

- Efficiency of synthesis algorithms for reactive systems

Approach

- Exploitation of specific properties in the specification, namely: monotony in counting constraints
- Successive increase of constraint length
- Usage of retrieved knowledge of previous steps allows for incomplete automata construction

Results

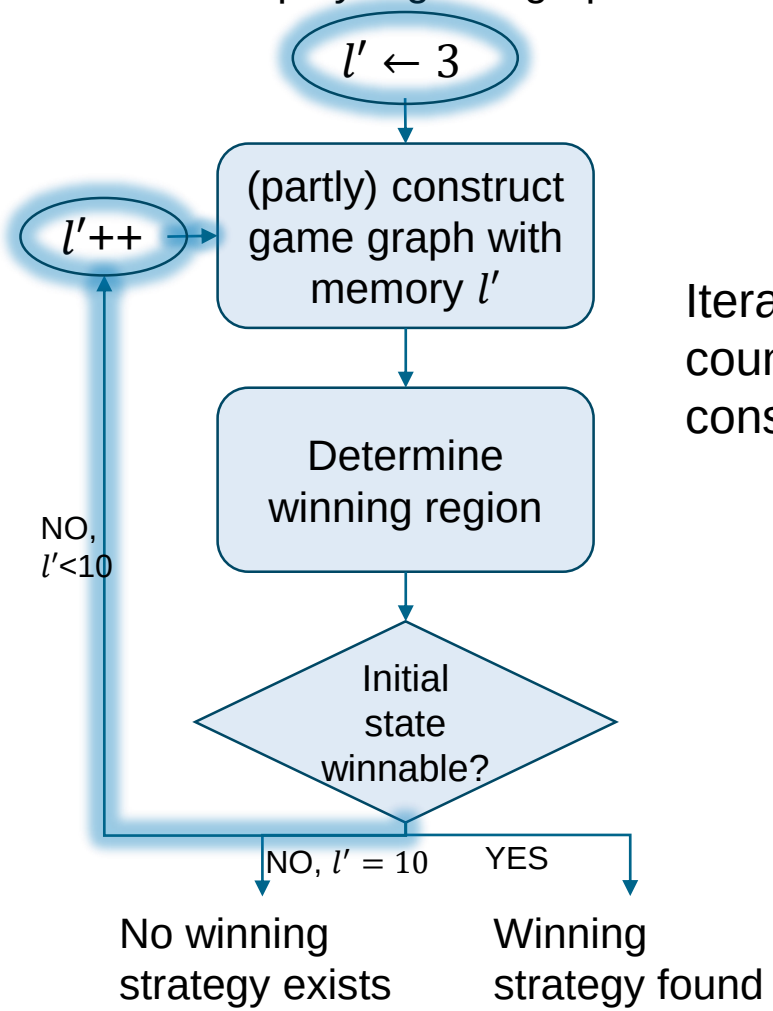
- Algorithm for synthesis with counting constraints
- Experimental results show promising savings in computational time & memory
- Future reseach directions determined

Thanks for your attention.
Questions are welcome!

BACKUP

Comparison for the Experiment

In: - EGO plays c at least 3 time in **10** turns.
- finite two-player game graph



Iteration over counting constraints

In: - EGO plays c at least 3 time in **8** turns.
- finite two-player game graph

Algorithm for comparison