Reachability and Büchi games

Cong Yu

February 15, 2022

Overview

Motivation & References

Motivation: Reachability and Büchi games are important in system verification and testing. Computing the winning set of Büchi games is a central problem in computer aided verification with a large number of applications.

References:

Robert McNaughton.
Infinite games played on finite graphs.
Ann. Pure Appl. Logic, 65(2):149–184, 1993.

Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors.
Automata Logics, and Infinite Games: A Guide to Current Research.
Springer-Verlag, Berlin, Heidelberg, 2002.

Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan.
Deciding parity games in quasipolynomial time.
In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 252–263, New York, NY, USA, 2017. Association for Computing Machinery.

Hans L. Bodlaender, Michael J. Dinneen, and Bakhadyr Khoussainov.
On game-theoretic models of networks.
In Peter Eades and Tadao Takaoka, editors, *Algorithms and Computation, 12th International Symposium, ISAAC 2001, Christchurch, New Zealand, December 19-21, 2001, Proceedings*, volume 2223 of *Lecture Notes in Computer Science*, pages 550–561. Springer, 2001.

Reachability Game

A reachability game is a 2-player (namely P0 and P1) game on a directed finite graph.

Game graph: directed graph $G(\{V_0 \cup V_1\}, E)$.($\{V_0, V_1\}$is a partition of V)

Target set: target set is $T \subseteq \{V_0 \cup V_1\}$.

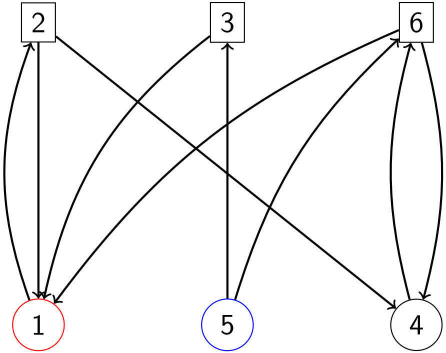
A play P is a (finite or infinite) path in the game graph beginning at the initial vertex s. If $v \in V_0$, P0 moves along an outgoing edge of v. Otherwise, P1 takes the move.

Definition of winning: P0 wins if $T \cap P \neq \emptyset$, otherwise P1 wins.

Memoryless strategy: a strategy for P0 is a mapping $\alpha : V_0 \rightarrow V$ that defines how P0 should extend the current play.

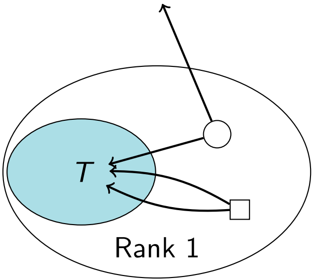
Example for Reachability Game

Rectangle vertices are in V_1, circles are in V_0;
Vertices in T are red, the initial vertex s is blue.



A winning play for P0 is $(5, 3, 1)$

Algorithm for Reachability Game



- if s is in T, P0 wins; 0 moves are needed for P0 to win.
- if $s \in V_0$ and s has at least one outgoing edge to $u \in T$, P0 wins in one step;
- if $s \in V_1$ and all of s's outgoing edges go to $u \in T$, P0 wins in one step;

Algorithm for Reachability Game

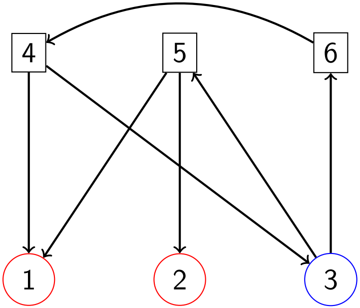
We defined Rank 0 and Rank 1 already, now we define Rank i.

An unranked vertex v now gets Rank i:
 if $v \in V_0$ and there is an edge $e(v, u) \quad u \in R_{i-1}$;
 if $v \in V_1$ and for every edge $e(v, u)$ we have $u \in \bigcup_{j=0}^{i-1} R_j$;

$R_i := \{v \in V|$ P0 can force a visit from v to a vertex in T in i steps$\}$

Define Reachability set of T for P0, $Reach(T, 0) := \bigcup_{i=1}^{n-1} R_i$

Algorithm for Reachability Game



- $R_0 = \{1, 2\}$;
- $R_1 = \{5\}$;
- $R_2 = \{3\}$;
- $R_3 = \{4\}$;
- $R_4 = \{6\}$;

For simplicity, denote $u \in R_k$ by Rank[u]=k.

An O(m) Algorithm for Reachability Game

Algorithm 1: Reachability for P0

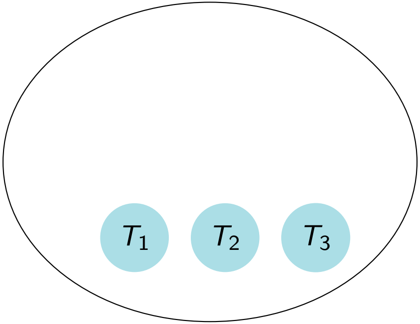
Data: game graph G, target set T
Result: Rank[$|V|$]

```
Q:= an empty queue;
Rank[|V|],count[|V|]:= all 0s array;
Q.push(T);
while Q is not empty do
    u:=Q.front,Q.pop;
    for e(v, u) ∈ E do
        if v ∈ V_0 and v has not been visited then
            Rank[v]:=Rank[u]+1; Q.push({v})
        else if v ∈ V_1 then
            count[v]:=count[v]+1;
            if count[v]=Out Degree of v then  Rank[v]:=Rank[u]+1; Q.push({v}) ;
        end
    end
end
```

Every edge is used at most once.

Type

T_1, T_2, ..., T_k are disjoint subsets of V, now we want to compute Reachability of each one of them.



Definition A type of vertex x is a tuple $(y_1, \ldots, y_k)$, where each $y_i \in \{0, 1\}$, such that $y_i = 1$ iff x is in $Reach(T_i, 0)$.

Compute Types

- Run reachability algorithm for every T_i, $O(km)$;
- Compute simultaneously.
- Can it be done in linear or nearly linear time?

Minimum Base

The minimum base of T is the minimum subset of T which can generate the same Reachability set as T.

Computing the minimum base is NP-hard.

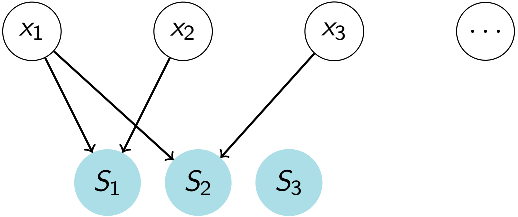
Set cover problem: Given a set S of n elements, a collection $S_1, S_2, ..., S_m$ of subsets of S, and a number K, does there exists a collection of at most k of these sets whose union is equal to all of S.

Minimum Base

Proof:
We prove that the decision problem for minimum base is NP-Complete.
The decision problem L is the following: Can we find a base with at most k vertices?

1. L is in NP.
2. set cover problem(which is NP-Complete) can be reduced to L in polynomial time.
   - Construct a Reachability game graph $G(V_0, E)$. There are m vertices in T representing m subsets in set cover problem, n vertices not in T representing n elements in S.
   - If subset S_i contains element x_j, connect an edge from vertex representing S_i to vertex representing x_j in T.

Minimum Base



$S_1 = \{x_1, x_2\}$
$S_2 = \{x_1, x_3\}$

So L is NP-Complete. The minimum base problem is NP-Hard.

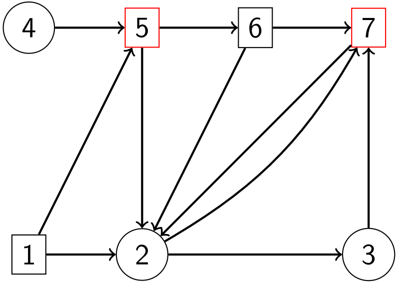
Büchi Game

Definition A **Büchi game** is a game $\mathcal{G} = (G, s, T)$ where G is the Reachability game graph, s is the initial vertex, $T \subseteq V$ is the target set as in Reachability game.

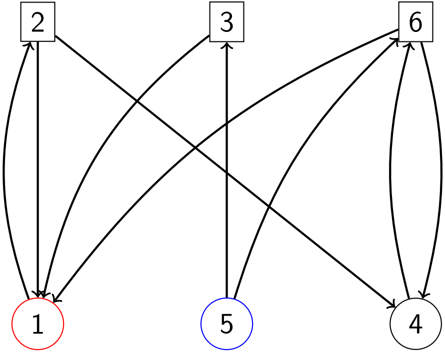
Play: The definition of play in Büchi Game is the same as in Reachability game.

Definition of winning: We assume the play P is infinite here. If there exists infinitely many vertices $v \in T$ in P, P0 wins. Otherwise P1 wins.

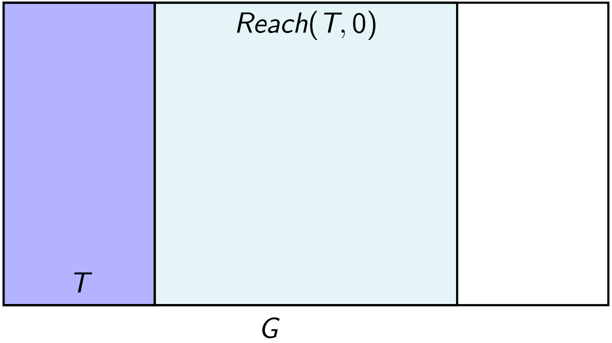
Example for Büchi Game



P0 is always winning on this game graph.



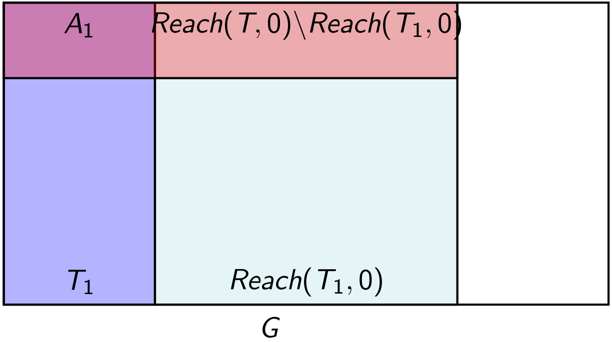
Algorithm for Büchi Game 1



If $v \notin Reach(T, 0) \cup T$, v can not reach T, P0 will lose.

Some vertices in T can not reach $Reach(T, 0) \cup T$, P0 will also lose on these vertices.

Algorithm for Büchi Game 1

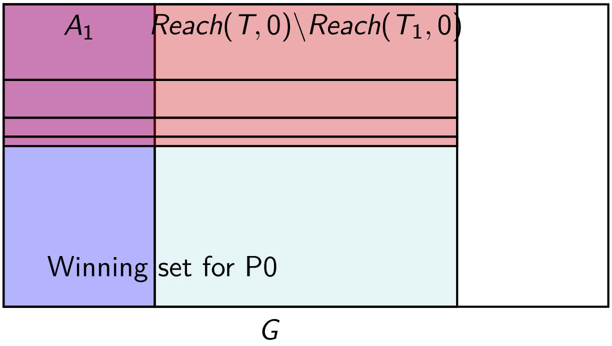


$A_1 = \{v \in T | v$ can't reach $T \cup Reach(T,0)\}$

Some vertices in T_1 can only reach $Reach(T,0) \setminus Reach(T_1,0)$

We find $A_2 = \{v \in T_1 | v$ can't reach $T_1 \cup Reach(T_1,0)\}$

Algorithm for Büchi Game 1



We repeat this process until T_k does not shrink.

The remaining part of $T_k \cup Reach(T_k, 0)$ is the winning set for P0.

Algorithm for Büchi Game 1

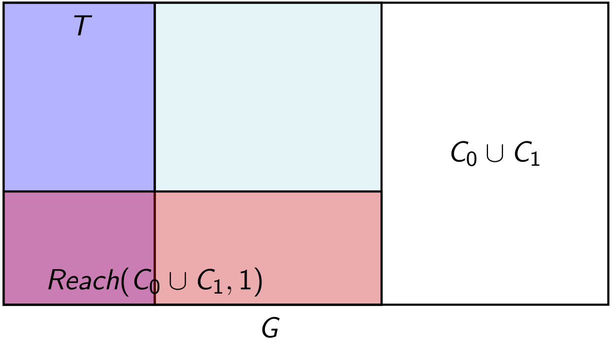
- How to find A_1

$$\begin{aligned} A_1 &= \{v \in T | v \text{ can't reach } T \cup Reach(T,0)\} \\ &= \{v \in T | v \text{ can only reach } V \backslash \{T \cup Reach(T,0)\}\} \\ &= Reach(V \backslash \{T \cup Reach(T,0)\}, 1) \end{aligned}$$

- Time complexity
  $O(m)$ to find A_i, at most $O(n)$ times. Worst-case $O(nm)$.

  Can it be done in nearly linear time?

Algorithm for Büchi Game 2

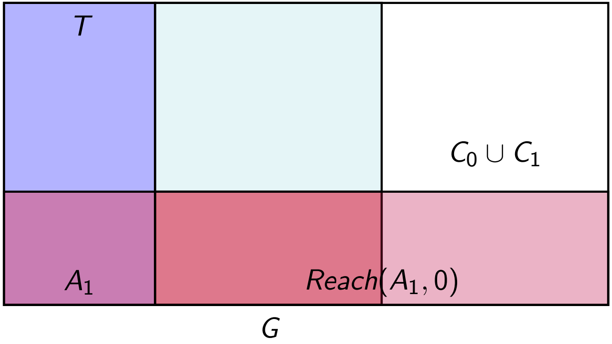


Compute C_0 and C_1.

C_0 is a set of vertices in $V_0 \setminus T$ having all outgoing edges to vertices in $V \setminus T$.
C_1 is a set of vertices in $V_1 \setminus T$ having an outgoing edge to vertices in $V \setminus T$.

Compute $Reach(C_0 \cup C_1, 1)$

Algorithm for Büchi Game 2

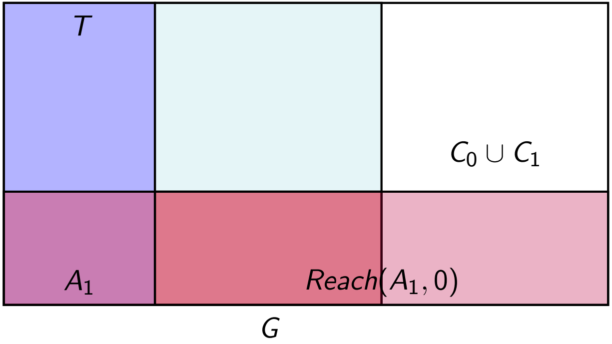


Some vertices in $Reach(C_0 \cup C_1, 1)$ can reach A_1.

Compute $Reach(A_1, 0)$.

$\{C_0 \cup C_1\} \backslash Reach(A_1, 0)$ is the set of vertices which can't reach T.

Algorithm for Büchi Game 2



$S = \{C_0 \cup C_1\} \backslash Reach(A_1, 0)$ is the same as $V \backslash \{T \cup Reach(T)\}$ in Algorithm 1.

Then we can compute $Reach(S, 1)$ to delete some losing vertices for P0 in T.

Repeat the same process on $G \backslash \{T \backslash Reach(S, 1)\}$

Algorithm for Büchi Game 2

- Time complexity
  $O(m)$ to find S, at most $O(n)$ times. Worst-case $O(nm)$.

- Advantage
  Algorithm 2 is preferable to algorithm 1 when $C_0 \cup C_1$ is small, algorithm 1 is preferable when T is small.

The End