

COMP3630 / COMP6363

*week 11: **Alternating Time***

This Chapter is not covered in HMU

*slides created by:* Dirk Pattinson, based on material by Peter Hoefner and Rob van Glabbeek; with improvements by Pascal Bercher

*convenor & lecturer:* Pascal Bercher

**The Australian National University**

Semester 1, 2023

## Content of this Chapter

- Alternating Turing Machines (ATMs)
- The complexity class **AP**
- **AP** vs. **PSPACE**

# The Geography Game

**Rules of Geography** given a designated starting city (e.g. Londonu)

- ① Player 1 names a city that begins with the last letter of the designated city (e.g. Newcastlee) and makes this the designated city.
- ② Player 2 names a city that begins with the last letter of the city named by player 2 (e.g. Edinburghh) and makes this the designated city, continue with rule 1.

**Winning Conditions.**

# The Geography Game

**Rules of Geography** given a designated starting city (e.g. Londonu)

- ① Player 1 names a city that begins with the last letter of the designated city (e.g. Newcastlee) and makes this the designated city.
- ② Player 2 names a city that begins with the last letter of the city named by player 2 (e.g. Edinburghh) and makes this the designated city, continue with rule 1.

**Winning Conditions.**

- The game is lost by the player that cannot name a city and won by the other player.

# The Geography Game

**Rules of Geography** given a designated starting city (e.g. London<sub>u</sub>)

- ① Player 1 names a city that begins with the last letter of the designated city (e.g. Newcastle<sub>u</sub>) and makes this the designated city.
- ② Player 2 names a city that begins with the last letter of the city named by player 2 (e.g. Edinburgh<sub>h</sub>) and makes this the designated city, continue with rule 1.

## Winning Conditions.

- The game is lost by the player that cannot name a city and won by the other player.

## Question.

*Does Player 1 have a winning strategy (i.e. can always win irrespective of the moves of player one)?*

(In “reality” we have partial knowledge but a hypothesis about what the other player knows (epistemic reasoning). Here we assume full knowledge (full observability.))

# The Proof Game

## Background.

- A formula  $A$  is provable if there is a proof rule with conclusion  $A$ , such that all its premisses are provable (e.g.  $\frac{B \rightarrow A \quad B}{A}$ )

# The Proof Game

## Background.

- A formula  $A$  is provable if there is a proof rule with conclusion  $A$ , such that all its premisses are provable (e.g.  $\frac{B \rightarrow A}{A} B$ )

**Rules of the Proof Game** for a given designated formula  $A_0$ :

- ① Player 1 chooses a proof rule  $\frac{A_1 \dots A_n}{A_0}$  whose conclusion is the designated formula.
- ② Player 2 chooses a premise  $A_i$  of the rule, and makes  $A_i$  the designated formula, continue with rule 1.

## Winning conditions.

# The Proof Game

## Background.

- A formula  $A$  is provable if there is a proof rule with conclusion  $A$ , such that all its premisses are provable (e.g.  $\frac{B \rightarrow A}{A} B$ )

**Rules of the Proof Game** for a given designated formula  $A_0$ :

- ① Player 1 chooses a proof rule  $\frac{A_1 \dots A_n}{A_0}$  whose conclusion is the designated formula.
- ② Player 2 chooses a premise  $A_i$  of the rule, and makes  $A_i$  the designated formula, continue with rule 1.

## Winning conditions.

- the player who cannot move loses the game
- infinite plays are lost by Player 1



# The Proof Game

## Background.

- A formula  $A$  is provable if there is a proof rule with conclusion  $A$ , such that all its premisses are provable (e.g.  $\frac{B \rightarrow A}{A} B$ )

**Rules of the Proof Game** for a given designated formula  $A_0$ :

- 1 Player 1 chooses a proof rule  $\frac{A_1 \dots A_n}{A_0}$  whose conclusion is the designated formula.
- 2 Player 2 chooses a premise  $A_i$  of the rule, and makes  $A_i$  the designated formula, continue with rule 1.

## Winning conditions.

- the player who cannot move loses the game
- infinite plays are lost by Player 1

## Question.

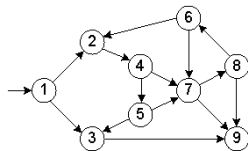
*Does Player 1 have a winning strategy (i.e. can always win irrespective of the moves of player one) so that  $A$  is provable?*

# The Generalised Geography Game

From Geography to Generalised Geography: Replace cities with directed graph:

## Rules.

- One node is always a designated node;
- Player 1 chooses a successor of the designated node which is the new designated node.
- Player 2 chooses a successor of the designated node which is the new designated node, continue with rule 1.



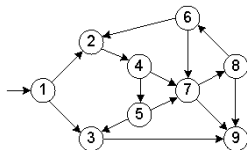
## Winning Conditions.

# The Generalised Geography Game

From Geography to Generalised Geography: Replace cities with directed graph:

## Rules.

- One node is always a designated node;
- Player 1 chooses a successor of the designated node which is the new designated node.
- Player 2 chooses a successor of the designated node which is the new designated node, continue with rule 1.



## Winning Conditions.

- who cannot move, loses
- Player 2 wins infinite plays

## Question.

*What is the complexity that – given graph  $G$  with designated initial node – of determining whether Player 1 has a winning strategy?*

## Problem Reductions between these Games

### From Geography to Generalised Geography.

Construct a graph where:

- the nodes are the names of cities
- there is an edge between city 1 and city 2 if the name of city 2 begins with the last letter of the name of city 1

## Problem Reductions between these Games

### From Geography to Generalised Geography.

Construct a graph where:

- the nodes are the names of cities
- there is an edge between city 1 and city 2 if the name of city 2 begins with the last letter of the name of city 1

### From Proof to Generalised Geography.

Construct a graph where:

- nodes are either formulae, or proof rules
- there is an edge between a formula node  $A$  and a proof rule node  $\frac{A_1 \dots A_n}{A_0}$  if  $A = A_0$
- there is an edge between a proof rule node  $\frac{A_1 \dots A_n}{A_0}$  and a formula node  $A$  if  $A = A_i$ , for some  $1 \leq i \leq n$ .

# Winning Strategies (for any 2-Player Game!)

**For Player 1 to win** from starting node  $n$ :

- there exists a move such that for all moves of player 2 to node  $n'$  ...
- Player 1 has a winning strategy from node  $n'$

**Pattern** for winning strategy:

- existential choice for player 1
- universal choice for player 2

## Recap: Non-deterministic Machines

### Complexity Class NP.

Have non-deterministic machine

- where every run takes at most polynomially many steps
- there exists an accepting sequence of IDs

### Complexity Class co-NP.

Have non-deterministic machine

- where every run takes at most polynomially many steps
- every sequence of IDs is accepting

**Alternating** Turing machines combine existential and universal runs

# Alternating Turing Machines

**Definition.** An alternating Turing machine is a non-deterministic Turing machine  $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$  where additionally  $Q = Q_e \cup Q_u$  is partitioned into a set of  $Q_e$  of existential states and  $Q_u$  of universal states.

## Instantaneous Descriptions (IDs)

- are defined as for non-deterministic machines, and contain tape content, head position, and state
- the transition relation  $I \vdash J$  between IDs is defined as for non-deterministic machines
- an ID is existential if the state is existential, and universal, if the state is universal.

Q. What about acceptance ... ?



## Acceptance Conditions

**Informally.** An ATM  $M$  accepts string  $w$  if there is a finite tree whose nodes are IDs and

- the root node is the initial ID ( $w$  on tape, state  $q_0$ ),
- every existential ID  $E$  has (exactly) one child  $J$  in the tree with  $E \vdash J$
- every universal ID  $U$  has all IDs  $J$  with  $U \vdash J$  as children, and
- all leaf nodes are universal (this implies there are no outgoing transitions).

## Acceptance Conditions

**Informally.** An ATM  $M$  accepts string  $w$  if there is a finite tree whose nodes are IDs and

- the root node is the initial ID ( $w$  on tape, state  $q_0$ ),
- every existential ID  $E$  has (exactly) one child  $J$  in the tree with  $E \vdash J$
- every universal ID  $U$  has all IDs  $J$  with  $U \vdash J$  as children, and
- all leaf nodes are universal (this implies there are no outgoing transitions).

Thus,  $L(M) = \{w : \text{There exists a tree as above with root } q_0 w \}$

## Acceptance Conditions

**Informally.** An ATM  $M$  accepts string  $w$  if there is a finite tree whose nodes are IDs and

- the root node is the initial ID ( $w$  on tape, state  $q_0$ ),
- every existential ID  $E$  has (exactly) one child  $J$  in the tree with  $E \vdash J$
- every universal ID  $U$  has all IDs  $J$  with  $U \vdash J$  as children, and
- all leaf nodes are universal (this implies there are no outgoing transitions).

Thus,  $L(M) = \{w : \text{There exists a tree as above with root } q_0 w \}$

### What about accepting states?

- We don't need/use them! We keep  $F$  for compatibility with the standard definition.

## Acceptance Conditions

**Informally.** An ATM  $M$  accepts string  $w$  if there is a finite tree whose nodes are IDs and

- the root node is the initial ID ( $w$  on tape, state  $q_0$ ),
- every existential ID  $E$  has (exactly) one child  $J$  in the tree with  $E \vdash J$
- every universal ID  $U$  has all IDs  $J$  with  $U \vdash J$  as children, and
- all leaf nodes are universal (this implies there are no outgoing transitions).

Thus,  $L(M) = \{w : \text{There exists a tree as above with root } q_0 w \}$

### What about accepting states?

- We don't need/use them! We keep  $F$  for compatibility with the standard definition.
- However, if we had acceptance states, then
  - An existential ID with no successors would never be accepting.
  - A universal ID with no successors would be accepting.

## Acceptance Conditions

**Informally.** An ATM  $M$  accepts string  $w$  if there is a finite tree whose nodes are IDs and

- the root node is the initial ID ( $w$  on tape, state  $q_0$ ),
- every existential ID  $E$  has (exactly) one child  $J$  in the tree with  $E \vdash J$
- every universal ID  $U$  has all IDs  $J$  with  $U \vdash J$  as children, and
- all leaf nodes are universal (this implies there are no outgoing transitions).

Thus,  $L(M) = \{w : \text{There exists a tree as above with root } q_0 w \}$

### What about accepting states?

- We don't need/use them! We keep  $F$  for compatibility with the standard definition.
- However, if we had acceptance states, then
  - An existential ID with no successors would never be accepting.
  - A universal ID with no successors would be accepting.
  - Note that now IDs are accepting/rejecting, not states.
  - Each ID in a tree as above would be accepting.

## Acceptance Conditions

**Informally.** An ATM  $M$  accepts string  $w$  if there is a finite tree whose nodes are IDs and

- the root node is the initial ID ( $w$  on tape, state  $q_0$ ),
- every existential ID  $E$  has (exactly) one child  $J$  in the tree with  $E \vdash J$
- every universal ID  $U$  has all IDs  $J$  with  $U \vdash J$  as children, and
- all leaf nodes are universal (this implies there are no outgoing transitions).

Thus,  $L(M) = \{w : \text{There exists a tree as above with root } q_0 w \}$

### What about accepting states?

- We don't need/use them! We keep  $F$  for compatibility with the standard definition.
- However, if we had acceptance states, then
  - An existential ID with no successors would never be accepting.
  - A universal ID with no successors would be accepting.
  - Note that now IDs are accepting/rejecting, not states.
  - Each ID in a tree as above would be accepting.

### How about loops?

- We require our tree to be finite, so we can't loop forever.

## Acceptance Conditions

**Informally.** An ATM  $M$  accepts string  $w$  if there is a finite tree whose nodes are IDs and

- the root node is the initial ID ( $w$  on tape, state  $q_0$ ),
- every existential ID  $E$  has (exactly) one child  $J$  in the tree with  $E \vdash J$
- every universal ID  $U$  has all IDs  $J$  with  $U \vdash J$  as children, and
- all leaf nodes are universal (this implies there are no outgoing transitions).

Thus,  $L(M) = \{w : \text{There exists a tree as above with root } q_0 w \}$

### What about accepting states?

- We don't need/use them! We keep  $F$  for compatibility with the standard definition.
- However, if we had acceptance states, then
  - An existential ID with no successors would never be accepting.
  - A universal ID with no successors would be accepting.
  - Note that now IDs are accepting/rejecting, not states.
  - Each ID in a tree as above would be accepting.

### How about loops?

- We require our tree to be finite, so we can't loop forever.
- The definition above does not require to "stick with decisions", i.e., any ID (both existential and universal) could occur several times. This is not a problem (since the tree is still finite), but we can cut out these "finite loops" by making decisions for the existential IDs that lead to the leafs earlier.

## Informal Example: Generalised Geography

### Solving via ATM.

- On tape: Graph and designated node.
- Two states,  $q_0$  (initial and existential) and  $q_1$  (universal)  
(Omitting intermediate states that are needed to change designated node.)
- From one state to another: replace designated node by successor in graph.

### Explanation.

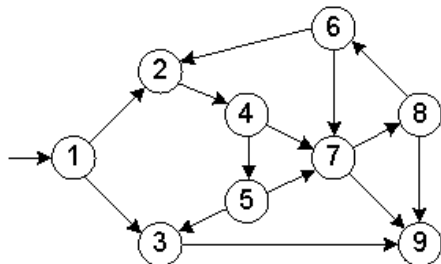
- IDs with  $q_0$  are the states where player 1 moves, and IDs with  $q_1$  are states where player 2 moves.
- If an ID with state  $q_0$  doesn't have outgoing transitions: Player 1 loses.
- If an ID with state  $q_1$  doesn't have outgoing transitions: Player 2 loses (as this is part of a winning strategy for player 1).

We'll re-visit this algorithm more formally in a few slides!



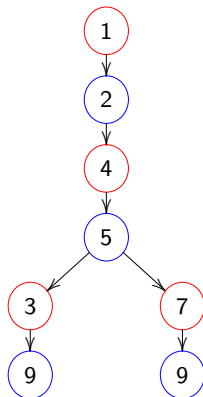
## Informal Example: Generalised Geography, cont'd

## Geography Graph.



- existential states are **red**, universal states are **blue**
- In general, existential states and universal states don't have to alternate!  
Here we have this since we use the ATM for solving a turn-taking 2-player game.

## Winning Strategy.



## First (ATM) Algorithm for Geography

```
Algorithm Geography (Graph G, start node n):  
  let cur = n;  
  forever do {  
    existentially guess (a successor node e of cur);  
    // if this is not possible, we don't accept  
  
    universally guess (a successor node u of e);  
    // if there are no successors, we accept  
  
    let cur := u;  
  }
```

### Comments.

- This shows (modulo a translation to TM) that Geography is solvable using an ATM.

# First (ATM) Algorithm for Geography

```
Algorithm Geography (Graph G, start node n):  
  let cur = n;  
  forever do {  
    existentially guess (a successor node e of cur);  
    // if this is not possible, we don't accept  
  
    universally guess (a successor node u of e);  
    // if there are no successors, we accept  
  
    let cur := u;  
  }
```

## Comments.

- This shows (modulo a translation to TM) that Geography is solvable using an ATM.
- However the number of steps that this ATM takes is possibly infinite if there are loops in the graph. But looping means we neither accept nor reject, but we know that the problem should be (is) decidable! (We'll revisit this Algorithm.)

## Restrictions of ATMs

**Definition.** An ATM is polytime bounded if there exists a polynomial  $p$  such that every sequence of IDs from an initial ID  $(q_0, w)$  is at most  $p(|w|)$  steps long.

The class **AP** of alternating polytime languages is the class of languages accepted by an ATM that is polytime bounded.

## Restrictions of ATMs

**Definition.** An ATM is polytime bounded if there exists a polynomial  $p$  such that every sequence of IDs from an initial ID  $(q_0, w)$  is at most  $p(|w|)$  steps long.

The class **AP** of alternating polytime languages is the class of languages accepted by an ATM that is polytime bounded.

### Observation.

- **NP**  $\subseteq$  **AP** (because we only need existential states)
- **co-NP**  $\subseteq$  **AP** (because we only need universal states)

## Restrictions of ATMs

**Definition.** An ATM is polytime bounded if there exists a polynomial  $p$  such that every sequence of IDs from an initial ID  $(q_0, w)$  is at most  $p(|w|)$  steps long.

The class **AP** of alternating polytime languages is the class of languages accepted by an ATM that is polytime bounded.

### Observation.

- **NP**  $\subseteq$  **AP** (because we only need existential states)
- **co-NP**  $\subseteq$  **AP** (because we only need universal states)
- Both will also follow directly because – spoiler – we are going to show **AP** = **PSPACE**, and both statements are known with regard to **PSPACE**.

## Restrictions of ATMs

**Definition.** An ATM is polytime bounded if there exists a polynomial  $p$  such that every sequence of IDs from an initial ID  $(q_0, w)$  is at most  $p(|w|)$  steps long.

The class **AP** of alternating polytime languages is the class of languages accepted by an ATM that is polytime bounded.

### Observation.

- **NP**  $\subseteq$  **AP** (because we only need existential states)
- **co-NP**  $\subseteq$  **AP** (because we only need universal states)
- Both will also follow directly because – spoiler – we are going to show **AP** = **PSPACE**, and both statements are known with regard to **PSPACE**.

### Reductions.

- Recall Theorem 10.1.7., week 7: If  $B$  is **NP**-hard and  $B \leq_P C$ , then

## Restrictions of ATMs

**Definition.** An ATM is polytime bounded if there exists a polynomial  $p$  such that every sequence of IDs from an initial ID  $(q_0, w)$  is at most  $p(|w|)$  steps long.

The class **AP** of alternating polytime languages is the class of languages accepted by an ATM that is polytime bounded.

### Observation.

- **NP**  $\subseteq$  **AP** (because we only need existential states)
- **co-NP**  $\subseteq$  **AP** (because we only need universal states)
- Both will also follow directly because – spoiler – we are going to show **AP** = **PSPACE**, and both statements are known with regard to **PSPACE**.

### Reductions.

- Recall Theorem 10.1.7., week 7: If  $B$  is **NP**-hard and  $B \leq_P C$ , then  $C$  is **NP**-hard.



## Restrictions of ATMs

**Definition.** An ATM is polytime bounded if there exists a polynomial  $p$  such that every sequence of IDs from an initial ID  $(q_0, w)$  is at most  $p(|w|)$  steps long.

The class **AP** of alternating polytime languages is the class of languages accepted by an ATM that is polytime bounded.

### Observation.

- **NP**  $\subseteq$  **AP** (because we only need existential states)
- **co-NP**  $\subseteq$  **AP** (because we only need universal states)
- Both will also follow directly because – spoiler – we are going to show **AP** = **PSPACE**, and both statements are known with regard to **PSPACE**.

### Reductions.

- Recall Theorem 10.1.7., week 7: If  $B$  is **NP**-hard and  $B \leq_P C$ , then  $C$  is **NP**-hard.
- Recall Theorem 10.1.5., week 7: If  $A \leq_P B$  and  $B \in \mathbf{P}$ , then

## Restrictions of ATMs

**Definition.** An ATM is polytime bounded if there exists a polynomial  $p$  such that every sequence of IDs from an initial ID  $(q_0, w)$  is at most  $p(|w|)$  steps long.

The class **AP** of alternating polytime languages is the class of languages accepted by an ATM that is polytime bounded.

### Observation.

- **NP**  $\subseteq$  **AP** (because we only need existential states)
- **co-NP**  $\subseteq$  **AP** (because we only need universal states)
- Both will also follow directly because – spoiler – we are going to show **AP** = **PSPACE**, and both statements are known with regard to **PSPACE**.

### Reductions.

- Recall Theorem 10.1.7., week 7: If  $B$  is **NP**-hard and  $B \leq_P C$ , then  $C$  is **NP**-hard.
- Recall Theorem 10.1.5., week 7: If  $A \leq_P B$  and  $B \in \mathbf{P}$ , then  $A \in \mathbf{P}$ .

## Restrictions of ATMs

**Definition.** An ATM is polytime bounded if there exists a polynomial  $p$  such that every sequence of IDs from an initial ID  $(q_0, w)$  is at most  $p(|w|)$  steps long.

The class **AP** of alternating polytime languages is the class of languages accepted by an ATM that is polytime bounded.

### Observation.

- **NP**  $\subseteq$  **AP** (because we only need existential states)
- **co-NP**  $\subseteq$  **AP** (because we only need universal states)
- Both will also follow directly because – spoiler – we are going to show **AP** = **PSPACE**, and both statements are known with regard to **PSPACE**.

### Reductions.

- Recall Theorem 10.1.7., week 7: If  $B$  is **NP**-hard and  $B \leq_P C$ , then  $C$  is **NP**-hard.
- Recall Theorem 10.1.5., week 7: If  $A \leq_P B$  and  $B \in \mathbf{P}$ , then  $A \in \mathbf{P}$ .
- Now: If  $L \leq_P L'$  and  $L' \in \mathbf{AP}$ , then

## Restrictions of ATMs

**Definition.** An ATM is polytime bounded if there exists a polynomial  $p$  such that every sequence of IDs from an initial ID  $(q_0, w)$  is at most  $p(|w|)$  steps long.

The class **AP** of alternating polytime languages is the class of languages accepted by an ATM that is polytime bounded.

### Observation.

- **NP**  $\subseteq$  **AP** (because we only need existential states)
- **co-NP**  $\subseteq$  **AP** (because we only need universal states)
- Both will also follow directly because – spoiler – we are going to show **AP** = **PSPACE**, and both statements are known with regard to **PSPACE**.

### Reductions.

- Recall Theorem 10.1.7., week 7: If  $B$  is **NP**-hard and  $B \leq_P C$ , then  $C$  is **NP**-hard.
- Recall Theorem 10.1.5., week 7: If  $A \leq_P B$  and  $B \in \mathbf{P}$ , then  $A \in \mathbf{P}$ .
- Now: If  $L \leq_P L'$  and  $L' \in \mathbf{AP}$ , then  $L \in \mathbf{AP}$ .

## Example Revisited: Geography

### Earlier Algorithm.

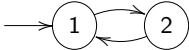
```

Algorithm Geography (Graph G, start node n):
  let cur = n;
  forever do {
    existentially guess (a successor node e of cur);
    // if this is not possible, we don't accept

    universally guess (a successor node u of e);
    // if there are none, we accept

    let cur := u;
  }

```

- not necessarily terminating, e.g.  (assume “fitting” transitions)
- let alone in polynomially many steps!

# Geography, Terminating

**Idea.** Universal nodes don't need to repeat. (I.e., the Ex.-player doesn't play into loops.)  
(Existential nodes also don't have to, but it doesn't hurt either!)

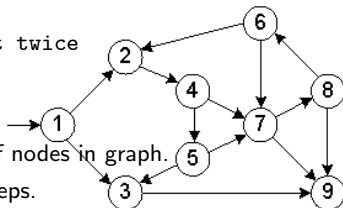
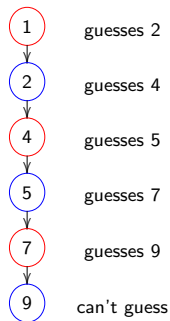
Algorithm Geography2 (Graph G, start node cur):

```

let seen := { cur };
forever do { // Player 1:
  existentially guess (cur := unseen successor of cur)
  // if this fails, we terminate and don't accept

  // Player 2:
  universally guess (cur := successor of cur);
  // if this fails, we terminate and accept

  seen := seen 'union' { cur } // never visit twice
}
  
```



## Geography in AP.

- Branches of tree at most twice as long as number of nodes in graph.
- Every computation path takes polynomially many steps.

## AP vs co-AP

**Observation.** Given polytime bounded ATM  $M$ , construct ATM  $M'$  by swapping existential and universal states

- then  $M'$  accepts  $w$  if and only if  $M$  rejects  $w$
- requires that all runs are terminating

**Corollary.**  $\text{co-AP} = \text{AP}$  (Again, this also follows from  $\text{AP} = \text{PSPACE}$ )

**Example.** What are the strings accepted by the TM and its dual version below, where \* indicates any letter?



## AP vs co-AP

**Observation.** Given polytime bounded ATM  $M$ , construct ATM  $M'$  by swapping existential and universal states

- then  $M'$  accepts  $w$  if and only if  $M$  rejects  $w$
- requires that all runs are terminating

**Corollary.**  $\text{co-AP} = \text{AP}$  (Again, this also follows from  $\text{AP} = \text{PSPACE}$ )

**Example.** What are the strings accepted by the TM and its dual version below, where  $*$  indicates any letter?



Their languages, in both cases, is  $\emptyset$ , because we never can't generate the desired proof tree with universal leaf nodes.



## AP vs co-AP

**Observation.** Given polytime bounded ATM  $M$ , construct ATM  $M'$  by swapping existential and universal states

- then  $M'$  accepts  $w$  if and only if  $M$  rejects  $w$
- requires that all runs are terminating

**Corollary.**  $\text{co-AP} = \text{AP}$  (Again, this also follows from  $\text{AP} = \text{PSPACE}$ )

**Example.** What are the strings accepted by the TM and its dual version below, where  $*$  indicates any letter?



Their languages, in both cases, is  $\emptyset$ , because we never can't generate the desired proof tree with universal leaf nodes.

**Exercise.** Construct a simple ATM (with 2 states) that terminates on all runs and check above's claim. E.g., one that only runs to the right until the first blank is found and changes the input word.

# QBF Revisited

**Idea.**  $\exists \rightsquigarrow$  existential guess,  $\forall \rightsquigarrow$  universal guess

Algorithm evalqbf (formula A):

```

case A of {
  A_1 OR A_2: if (evalqbf A_1) = 1 then 1 else evalqnf(A_2)
  A_1 AND A_2: if (evalqbf A_1) = 0 then 0 else evalqbf(A_2)
  NOT A_1 : return 1 - (evalqnf A_1)
  exists x A : existentially guess v in {0, 1};
                evalqbf A [ x := v]
  forall x A : universally guess v in {0, 1};
                evalqbf A [ x := v]
}

```

where  $A [x := v]$  replaces all free occurrences of  $x$  in  $A$  with  $v$ .

**Theorem.**

- QBF is in **AP** (by algorithm above)
- **PSPACE**  $\subseteq$  **AP** (as QBF is **PSPACE**-hard)

# From AP to PSPACE

**Theorem.**  $\text{AP} \subseteq \text{PSPACE}$ .

**Proof (Idea).** Depth-first search simulates ATM  $M$  on standard TM.

Algorithm `ATMAccept (ATM-ID I)`:

```

if (I is existential) {
  let accept := false;
  foreach J with I |- J { accept := accept OR ATMAccept(J); }
  return (accept);
} else if (I is universal) {
  let accept := true;
  foreach J with I |- J { accept := accept AND ATMAccept(J); }
  return (accept);
}

```

For polynomial bound  $p$  and input of length  $n$ :

- recursion depth is polynomial as  $M$  is **AP**
- argument in recursive calls is of size  $O(p(n))$

So space in  $O(p^2(n))$ .

## Why **PSPACE** is “harder” than **NP**

- True **NP**-complete instances can (at least) be easily verified:  
Provide witness (e.g., accepting-ID-path of NTM).  
Has polynomial length and can be verified in polynomial time.
- Example: Is a SAT formula satisfiable?  
Verifier can check correctness of variable assignment in polytime.

## Why **PSPACE** is “harder” than **NP**

- True **NP**-complete instances can (at least) be easily verified:  
Provide witness (e.g., accepting-ID-path of NTM).  
Has polynomial length and can be verified in polynomial time.
- Example: Is a SAT formula satisfiable?  
Verifier can check correctness of variable assignment in polytime.
- True **PSPACE**-complete problems can (probably) not be easily verified:  
Prover even of unlimited power cannot convince poly-time verifier that some language is in some class (if **PSPACE**  $\neq$  **NP**). (Because **PSPACE**  $\neq$  **NP** directly, due to our central corollary, excludes a polytime witness.)

# What we know and don't know (Recap)

## Inclusions

- $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME}$
- but  $\mathbf{P} \neq \mathbf{EXPTIME}$ , but we don't know which inclusion(s) is (are) strict

# What we know and don't know (Recap)

## Inclusions

- $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME}$
- but  $\mathbf{P} \neq \mathbf{EXPTIME}$ , but we don't know which inclusion(s) is (are) strict

## Co-Classes.

- $\mathbf{co-PSPACE} = \mathbf{PSPACE} = \mathbf{NPSPACE} = \mathbf{co-NPSPACE}$
- $\mathbf{NP} \subseteq \mathbf{PSPACE}$  and  $\mathbf{co-NP} \subseteq \mathbf{PSPACE}$
- but we don't know whether  $\mathbf{NP} = \mathbf{co-NP}$  (however this would follow if  $\mathbf{P} = \mathbf{NP}$ )

# What we know and don't know (Recap)

## Inclusions

- $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME}$
- but  $\mathbf{P} \neq \mathbf{EXPTIME}$ , but we don't know which inclusion(s) is (are) strict

## Co-Classes.

- $\mathbf{co-PSPACE} = \mathbf{PSPACE} = \mathbf{NPSPACE} = \mathbf{co-NPSPACE}$
- $\mathbf{NP} \subseteq \mathbf{PSPACE}$  and  $\mathbf{co-NP} \subseteq \mathbf{PSPACE}$
- but we don't know whether  $\mathbf{NP} = \mathbf{co-NP}$  (however this would follow if  $\mathbf{P} = \mathbf{NP}$ )

## Equalities (from Today)

- $\mathbf{PSPACE} = \mathbf{AP}$