COMP3630 / COMP6363

week 10: **Alternating Time**

Not based on the book

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

convenor & lecturer: Pascal Bercher

The Australian National University

Semester 1, 2025

Content of this Chapter

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

Games

The Geography Game

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

1. Player 1 names a city that begins with the last letter of the designated city (e.g., Newcastle) and makes this the designated city (i.e., Newcastle).

The Geography Game

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

1. Player 1 names a city that begins with the last letter of the designated city (e.g., Newcastle) and makes this the designated city (i.e., Newcastle).
2. Player 2 names a city that begins with the last letter of the city named by player 1 (e.g., Edinburgh) and makes this the designated city (i.e., Edinburgh).

Continue with rule 1

Winning Conditions.

The Geography Game

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

1. Player 1 names a city that begins with the last letter of the designated city (e.g., Newcastle) and makes this the designated city (i.e., Newcastle).
2. Player 2 names a city that begins with the last letter of the city named by player 1 (e.g., Edinburgh) and makes this the designated city (i.e., Edinburgh).

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)

1. Player 1 names a city that begins with the last letter of the designated city (e.g., Newcastle) and makes this the designated city (i.e., Newcastle).
2. Player 2 names a city that begins with the last letter of the city named by player 1 (e.g., Edinburgh) and makes this the designated city (i.e., Edinburgh).

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 2)?

(In "reality" we have partial knowledge but a hypothesis about what the other player knows (epistemic reasoning). Here we assume full knowledge (i.e., 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. $\frac{B \to A \qquad B}{A}$)

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. $\frac{B \to A \qquad B}{A}$)

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

1. Player 1 chooses a proof rule $\frac{A_1 \quad \dots \quad A_n}{A_0}$ whose conclusion is the designated formula.

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. $\frac{B \to A \qquad B}{A}$)

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

1. Player 1 chooses a proof rule $\frac{A_1 \quad \dots \quad A_n}{A_0}$ 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. $\frac{B \to A \qquad B}{A}$)

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

1. Player 1 chooses a proof rule $\frac{A_1 \quad \dots \quad A_n}{A_0}$ 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. Why?

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. $\frac{B \rightarrow A \quad B}{A}$)

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

1. Player 1 chooses a proof rule $\frac{A_1 \quad \dots \quad A_n}{A_0}$ 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. Why?
  - player 1 can't move: player 2 picked a premise that can't be proved by player 1
  - player 2 can't move: the chosen premise is an axiom (with no premises)

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. $\frac{B \to A \quad B}{A}$)

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

1. Player 1 chooses a proof rule $\frac{A_1 \quad \dots \quad A_n}{A_0}$ 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. Why?
  - player 1 can't move: player 2 picked a premise that can't be proved by player 1
  - player 2 can't move: the chosen premise is an axiom (with no premises)
- infinite plays are lost by player 1. Why?

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. $\frac{B \to A \quad B}{A}$)

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

1. Player 1 chooses a proof rule $\frac{A_1 \quad \dots \quad A_n}{A_0}$ 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. Why?
  - player 1 can't move: player 2 picked a premise that can't be proved by player 1
  - player 2 can't move: the chosen premise is an axiom (with no premises)
- infinite plays are lost by player 1. Why?
  That means the proof does not end in axioms but is cyclic.

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. $\frac{B \to A \quad B}{A}$)

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

1. Player 1 chooses a proof rule $\frac{A_1 \quad \dots \quad A_n}{A_0}$ 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.

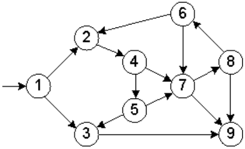
Winning conditions.

- the player who cannot move loses the game. Why?
  - player 1 can't move: player 2 picked a premise that can't be proved by player 1
  - player 2 can't move: the chosen premise is an axiom (with no premises)
- infinite plays are lost by player 1. Why?
  That means the proof does not end in axioms but is cyclic.

Question.
Does player 1 have a winning strategy?

The Generalized Geography Game

From Geography to Generalized Geography: Replace cities with directed graph:
The graph has a designated start start node.



Rules.

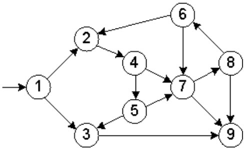
1. Player 1 chooses a successor of the designated node, which is the new designated node for player 2.
2. Player 2 chooses a successor of the designated node, which is the new designated node for player 1.

Continue with rule 1.

Winning Conditions.

The Generalized Geography Game

From Geography to Generalized Geography: Replace cities with directed graph:
The graph has a designated start start node.



Rules.

1. Player 1 chooses a successor of the designated node, which is the new designated node for player 2.
2. Player 2 chooses a successor of the designated node, which is the new designated node for player 1.

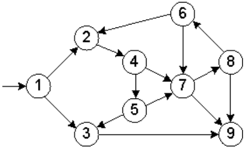
Continue with rule 1.

Winning Conditions.

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

The Generalized Geography Game

From Geography to Generalized Geography: Replace cities with directed graph:
The graph has a designated start start node.



Rules.

1. Player 1 chooses a successor of the designated node, which is the new designated node for player 2.
2. Player 2 chooses a successor of the designated node, which is the new designated node for player 1.

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

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

In conclusion: Generalized Geography is at least as hard as the other two problems.

Winning Strategies (for any 2-Player Game!)

Player 1 has a winning strategy from node n if:

- 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 (i.e., one has to work for player 1)
- universal "choice" for player 2 (i.e., all have to work for player 2, or, equivalently, (hence "choice"!) one chosen one has not to work for player 1)

Can we model such a strategy using non-deterministic TMs?

Winning Strategies (for any 2-Player Game!)

Player 1 has a winning strategy from node n if:

- 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 (i.e., one has to work for player 1)
- universal "choice" for player 2 (i.e., all have to work for player 2, or, equivalently, (hence "choice"!) one chosen one has not to work for player 1)

Can we model such a strategy using non-deterministic TMs? (Compare with the failed **NP** = **co-NP**) proof!

Winning Strategies (for any 2-Player Game!)

Player 1 has a winning strategy from node n if:

- 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 (i.e., one has to work for player 1)
- universal "choice" (hence "choice"!) for player 2 (i.e., all have to work for player 2, or, equivalently, one chosen one has not to work for player 1)

Can we model such a strategy using non-deterministic TMs? (Compare with the failed **NP** = **co-NP**) proof! We don't know. :) But not easily: It mixes **NP** with **co-NP**, so the acceptance criteria don't mix.

Winning Strategies (for any 2-Player Game!)

Player 1 has a winning strategy from node n if:

- 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 (i.e., one has to work for player 1)
- universal "choice" for player 2 (i.e., all have to work for player 2, or, equivalently, (hence "choice"!) one chosen one has not to work for player 1)

Can we model such a strategy using non-deterministic TMs? (Compare with the failed **NP** = **co-NP**) proof! We don't know. :) But not easily: It mixes **NP** with **co-NP**, so the acceptance criteria don't mix.

So, what's the solution? A more complex model for Turing Machines!

Alternating Turing Machines (ATMs)

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.

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,

- all as above, but
- that decides the complement of the problem
- this means that for yes-instances every sequence of IDs is accepting

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,

- all as above, but
- that decides the complement of the problem
- this means that for yes-instances every sequence of IDs is accepting

Alternating Turing machines combine existential and universal runs

Alternating Turing Machines

Definition. An Alternating Turing machine (ATM) 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.

Alternating Turing Machines

Definition. An Alternating Turing machine (ATM) 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 before
- the transition relation $I \vdash J$ between IDs is (also) defined as before
- an ID is existential if its state is existential, and universal if its state is universal.

Alternating Turing Machines

Definition. An Alternating Turing machine (ATM) 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 before
- the transition relation $I \vdash J$ between IDs is (also) defined as before
- an ID is existential if its state is existential, and universal if its state is universal.

Q. What about acceptance ...?

Acceptance Conditions

Informally. An ATM M accepts string w iff 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 iff 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, $L(M) = \{w \mid \text{ There exists a tree as above with root ID } q_0 w \}$

Acceptance Conditions

Informally. An ATM M accepts string w iff 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, $L(M) = \{w \mid \text{ There exists a tree as above with root ID } q_0 w\}$

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

What about accepting states?

- We don't need/use them! We keep F for compatibility with the standard definition.
- However, if we would have acceptance stati, 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 iff 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, $L(M) = \{w \mid \text{ There exists a tree as above with root ID } q_0 w \ \}$

What about accepting states?

- We don't need/use them! We keep F for compatibility with the standard definition.
- However, if we would have acceptance stati, 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. (How is that different?)

Acceptance Conditions

Informally. An ATM M accepts string w iff 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, $L(M) = \{w \mid \text{ There exists a tree as above with root ID } q_0 w \}$

What about accepting states?

- We don't need/use them! We keep F for compatibility with the standard definition.
- However, if we would have acceptance stati, 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. (How is that different?)
- Each ID in a tree as above would be accepting.

Acceptance Conditions

Informally. An ATM M accepts string w iff 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, $L(M) = \{w \mid \text{ There exists a tree as above with root ID } q_0 w \}$

What about accepting states?

- We don't need/use them! We keep F for compatibility with the standard definition.
- However, if we would have acceptance stati, 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. (How is that different?)
- Each ID in a tree as above would be accepting.

How about loops?

- We require our tree to be finite (not a graph!), so we can't loop forever.

Acceptance Conditions

Informally. An ATM M accepts string w iff 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, $L(M) = \{w \mid \text{ There exists a tree as above with root ID } q_0 w \}$

What about accepting states?

- We don't need/use them! We keep F for compatibility with the standard definition.
- However, if we would have acceptance stati, 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. (How is that different?)
- Each ID in a tree as above would be accepting.

How about loops?

- We require our tree to be finite (not a graph!), 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 could cut out these "detours" by making decisions for the existential IDs that lead to the leafs earlier (hence making it a deterministic policy).

Informal Example: Generalised Geography

Solving via ATM.

- On tape: Graph and designated node.
- Two states, q_0 (initial and existential) and q_1 (universal)
- From one state to another:
  - the player changes (that is exactly why the change states!)
  - replace designated node by successor in graph
  - we might need more existential states to encode changing the designated node according to the graph (it should be clear that deterministic TMs can do that).

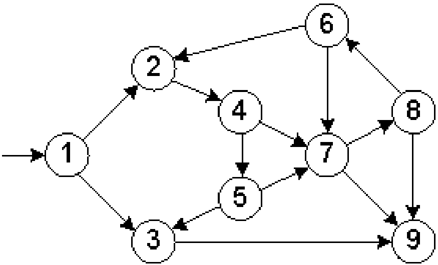
Explanation.

- IDs containing state q_0 are those where player 1 moves.
- IDs containing state q_1 are those where player 2 moves.
- If an ID containing state q_0 doesn't have outgoing transitions: player 1 loses.
- If an ID containing state q_1 doesn't have outgoing transitions: player 1 wins.

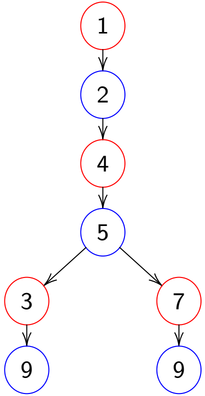
We'll re-visit this algorithm more formally in a few slides!

Informal Example: Generalised Geography, cont'd

Geography Graph.



Winning Strategy.



- 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.

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

    cur := u; }
```

Comments.

- This hints at Geography being solvable using an ATM (modulo translation to a NTM). Why just hinting at? What's missing?

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

    cur := u; }
```

Comments.

- This hints at Geography being solvable using an ATM (modulo translation to a NTM). Why just hinting at? What's missing?
- It's not a decider yet! It might loop forever if there are loops in the graph. (We'll revisit this Algorithm later.)

The class **AP**

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.

(We do not require the solution tree to be poly-bounded! Just ist maximal path!)

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_0w is at most $p(|w|)$ steps long.

(We do not require the solution tree to be poly-bounded! Just ist maximal path!)

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

Observation.

- **NP** $\subseteq$ **AP**. Why?

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.

(We do not require the solution tree to be poly-bounded! Just ist maximal path!)

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

Observation.

- **NP** $\subseteq$ **AP**. Why? Because we only need existential states; almost!

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.

(We do not require the solution tree to be poly-bounded! Just ist maximal path!)

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

Observation.

- **NP** $\subseteq$ **AP**. Why? Because we only need existential states; almost!
- **co-NP** $\subseteq$ **AP** Why?

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.

(We do not require the solution tree to be poly-bounded! Just ist maximal path!)

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

Observation.

- **NP** $\subseteq$ **AP**. Why? Because we only need existential states; almost!
- **co-NP** $\subseteq$ **AP** Why? Because we only need universal states; requires more reasoning!

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_0w is at most $p(|w|)$ steps long.

(We do not require the solution tree to be poly-bounded! Just ist maximal path!)

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

Observation.

- **NP** $\subseteq$ **AP**. Why? Because we only need existential states; almost!
- **co-NP** $\subseteq$ **AP** Why? Because we only need universal states; requires more reasoning!
- Both will also follow directly because – spoiler – we are going to show **AP** $=$ **PSPACE**, and both statements are known with regard to **PSPACE**.

  Wait, if these classes are identical, why did we even define this TM and class?!

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.

(We do not require the solution tree to be poly-bounded! Just ist maximal path!)

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

Observation.

- **NP** $\subseteq$ **AP**. Why? Because we only need existential states; almost!
- **co-NP** $\subseteq$ **AP** Why? Because we only need universal states; requires more reasoning!
- Both will also follow directly because – spoiler – we are going to show **AP** $=$ **PSPACE**, and both statements are known with regard to **PSPACE**.

  Wait, if these classes are identical, why did we even define this TM and class?!

  - Because we can! So, as always, we want to know whether that changes anything.
  - Because it might make some proofs easier. Think of games! (But much more.)

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.

(We do not require the solution tree to be poly-bounded! Just ist maximal path!)

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

Observation.

- **NP** $\subseteq$ **AP**. Why? Because we only need existential states; almost!
- **co-NP** $\subseteq$ **AP** Why? Because we only need universal states; requires more reasoning!
- Both will also follow directly because – spoiler – we are going to show **AP** $=$ **PSPACE**, and both statements are known with regard to **PSPACE**.

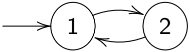
  Wait, if these classes are identical, why did we even define this TM and class?!
  - Because we can! So, as always, we want to know whether that changes anything.
  - Because it might make some proofs easier. Think of games! (But much more.)

Reductions/Hardness/Membership.
As always: defined as before.

Example Revisited: Geography

Earlier Algorithm.

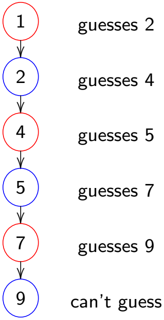
```
Algorithm Geography (Graph G, start node n):
  let cur = n;
  forever do {
    existentially guess (a successor node e of cur);
    // if this is not possible, we don't accept

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

    cur := u; }
```

- not necessarily terminating, e.g.,  (assume "fitting" transitions)

- let alone in polynomially many steps!

Geography, Terminating

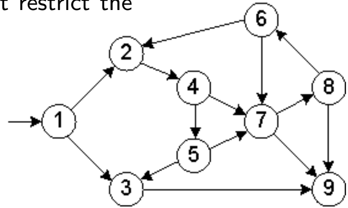
Idea. Universal nodes don't need to repeat:
I.e., the existential player doesn't repeat decisions.



Recap:

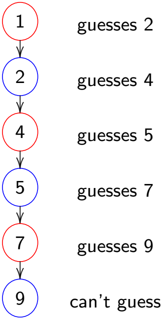
- Any algorithm needs to be a decider, i.e., have finite runtime.
- Any solution is a finite tree (not graph), i.e., can't have loops:
  - The existential player creating "some loops" doesn't hurt semantically: if there is a solution, eventually the right choice has to be made.
  - But we can make this "correct choice" right away, i.e., never repeat anything.

But ... Will this lead to termination although we do not restrict the moves by the universal player?



Geography, Terminating

Idea. Universal nodes don't need to repeat:
I.e., the existential player doesn't repeat decisions.

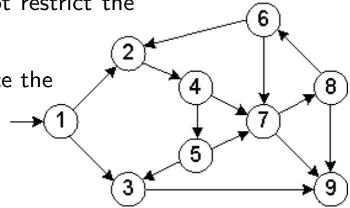


Recap:

- Any algorithm needs to be a decider, i.e., have finite runtime.
- Any solution is a finite tree (not graph), i.e., can't have loops:
  - The existential player creating "some loops" doesn't hurt semantically: if there is a solution, eventually the right choice has to be made.
  - But we can make this "correct choice" right away, i.e., never repeat anything.

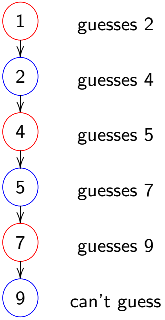
But ... Will this lead to termination although we do not restrict the moves by the universal player?

Yes! The length of each computation is bounded by twice the number of nodes in graph! Why?



Geography, Terminating

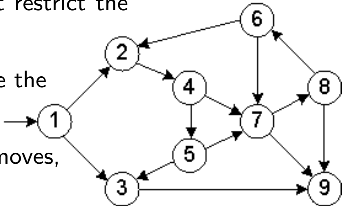
Idea. Universal nodes don't need to repeat:
I.e., the existential player doesn't repeat decisions.



Recap:

- Any algorithm needs to be a decider, i.e., have finite runtime.
- Any solution is a finite tree (not graph), i.e., can't have loops:
  - The existential player creating "some loops" doesn't hurt semantically: if there is a solution, eventually the right choice has to be made.
  - But we can make this "correct choice" right away, i.e., never repeat anything.

But ... Will this lead to termination although we do not restrict the moves by the universal player?



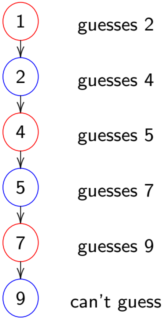
Yes! The length of each computation is bounded by twice the number of nodes in graph! Why?

The existential player can make at most $|V|$ (all nodes) moves, each followed by any (unrestricted) follow-up move.

Geography, Terminating

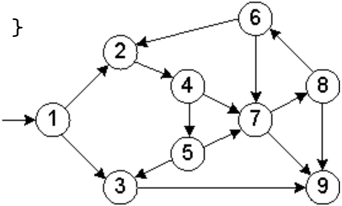
Idea. Universal nodes don't need to repeat:
I.e., the existential player doesn't repeat decisions.

```
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, representing reject

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

    seen := seen U { cur } // update seen nodes }
```



Geography is in AP:

- The algorithm takes only polynomially many steps.
- It recognizes the right language (although the code does not explicitly accept or reject anything).



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.

Corollary. **co-AP** = **AP** (Again, this also follows from **AP** = **PSPACE**)

Example. What are the strings accepted by the ATM 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.

Corollary. **co-AP** = **AP** (Again, this also follows from **AP** = **PSPACE**)

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

Note that the universal states always have a successor state (i.e., for all symbols), so it cannot accept anything.

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.

Corollary. **co-AP** = **AP** (Again, this also follows from **AP** = **PSPACE**)

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

Note that the universal states always have a successor state (i.e., for all symbols), so it cannot accept anything.

Exercise. Construct a more complex (but still simple) ATM that terminates on all runs and check above's claim.

Solving QBF via ATM

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

```
Algorithm evalqbf(formula A):
case A of {
  literal x or NOT x:  if true under the current assignment:
    enter a universal state without transitions
    else: enter an existential configuration without transitions
  A1 OR A2:     existentially choose i in {1, 2}, then evalqbf(Ai)
  A1 AND A2:    universally choose i in {1, 2}, then evalqbf(Ai)
  NOT A:        evalqbf_neg(A) // the dual of this machine
  exists x A:   existentially guess v in {0,1}, then evalqbf(A[x := v])
  forall x A:   universally guess v in {0,1}, then evalqbf(A[x := v])
}
```

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

Theorem w10.1

PSPACE $\subseteq$ **AP** (Solving QBF via ATM)

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

```
Algorithm evalqbf(formula A):
case A of {
  literal x or NOT x:  if true under the current assignment:
    enter a universal state without transitions
    else: enter an existential configuration without transitions
  A1 OR A2:     existentially choose i in {1, 2}, then evalqbf(Ai)
  A1 AND A2:    universally choose i in {1, 2}, then evalqbf(Ai)
  NOT A:        evalqbf_neg(A) // the dual of this machine
  exists x A:   existentially guess v in {0,1}, then evalqbf(A[x := v])
  forall x A:   universally guess v in {0,1}, then evalqbf(A[x := v])
}
```

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

Theorem w10.1

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

Simulating ATM on TM

Use the following algorithm for the initial configuration of an ATM.

```
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?;
  }
```

Observations:

- If the ATM M is an **AP** decider,
  - recursion depth is in $\mathcal{O}(p(n))$,
  - all IDs of size $\mathcal{O}(p(n))$.
- So, space of this DTM is in $\mathcal{O}(p^2(n))$.

PSPACE $\supseteq$ **AP** (Simulating ATM on TM)

Use the following algorithm for the initial configuration of an ATM.

```
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?;
  }
```

Observations:

- If the ATM M is an **AP** decider,
  - recursion depth is in $\mathcal{O}(p(n))$,
  - all IDs of size $\mathcal{O}(p(n))$.
- So, space of this DTM is in $\mathcal{O}(p^2(n))$.

Theorem w10.2

AP $\subseteq$ **PSPACE**.

Complexity Overviews

Theorem w10.3

AP = **PSPACE**.

We hence directly get: **AP** = **co-AP**

Complexity Overviews

Theorem w10.3

AP = **PSPACE**.

We hence directly get: **AP** = **co-AP**

Why is there no **NAP**?

Complexity Overviews

Theorem w10.3

AP = **PSPACE**.

We hence directly get: **AP** = **co-AP**

Why is there no **NAP**? It already subsumes non-determinism using its existential states!

So... Is any NTM an ATM? (Cf. slide from the beginning.)

Complexity Overviews

Theorem w10.3

AP = **PSPACE**.

We hence directly get: **AP** = **co-AP**

Why is there no **NAP**? It already subsumes non-determinism using its existential states!

So... Is any NTM an ATM? (Cf. slide from the beginning.)

- No! If we only have existential states, then

Complexity Overviews

Theorem w10.3

AP = **PSPACE**.

We hence directly get: **AP** = **co-AP**

Why is there no **NAP**? It already subsumes non-determinism using its existential states!

So... Is any NTM an ATM? (Cf. slide from the beginning.)

- No! If we only have existential states, then the ATM's language is empty!
- So, for each accepting state in the NTM, we need to introduce a universal state without outgoing transition.
- Thus, every NTM can trivially be considered an ATM.