CSCI 5451 Fall 2015

Week 13 Notes

Professor Ellen Gethner

November 9, 2015

NP-Completeness

- The story began in 1971 with Steve Cook's Theorem[1]...

[1] http://4mhz.de/cook.html

[2] Introduction to Algorithms: A Creative Approach by Udi Manber (available as a free download)

NP-Completeness

- The story began in 1971 with Steve Cook's Theorem[1]...

- **Preliminaries**[2]

---

[1] `http://4mhz.de/cook.html`

[2] Introduction to Algorithms: A Creative Approach by Udi Manber (available as a free download)

NP-Completeness

- The story began in 1971 with Steve Cook's Theorem[1]...

- **Preliminaries**[2]
  1. Rename *algorithm* as deterministic algorithm.

[1] http://4mhz.de/cook.html

[2] Introduction to Algorithms: A Creative Approach by Udi Manber (available as a free download)

NP-Completeness

- The story began in 1971 with Steve Cook's Theorem[1]...
- **Preliminaries**[2]
  1. Rename *algorithm* as deterministic algorithm.
  2. In all of the algorithms we have studied so far, there was each step was clear and determined.

[1] http://4mhz.de/cook.html

[2] Introduction to Algorithms: A Creative Approach by Udi Manber (available as a free download)

NP-Completeness

- The story began in 1971 with Steve Cook's Theorem[1]...

- **Preliminaries**[2]
  1. Rename *algorithm* as deterministic algorithm.
  2. In all of the algorithms we have studied so far, there was each step was clear and determined.
  3. The party is over!

[1] http://4mhz.de/cook.html

[2] Introduction to Algorithms: A Creative Approach by Udi Manber (available as a free download)

NP-Completeness

- The story began in 1971 with Steve Cook's Theorem[1]...

- **Preliminaries**[2]
  1. Rename *algorithm* as deterministic algorithm.
  2. In all of the algorithms we have studied so far, there was each step was clear and determined.
  3. The party is over!
  4. A nondeterministic algorithm has all the operations that a deterministic one has, but also has a new operation called nd-choice.

[1] http://4mhz.de/cook.html

[2] Introduction to Algorithms: A Creative Approach by Udi Manber (available as a free download)

NP-Completeness

- The story began in 1971 with Steve Cook's Theorem[1]...

- **Preliminaries**[2]
  1. Rename *algorithm* as deterministic algorithm.
  2. In all of the algorithms we have studied so far, there was each step was clear and determined.
  3. The party is over!
  4. A nondeterministic algorithm has all the operations that a deterministic one has, but also has a new operation called nd-choice.
  5. Operation nd-choice is associated with a fixed number of choices such that at each choice, the algorithm follows a different computation path.

[1] `http://4mhz.de/cook.html`

[2] Introduction to Algorithms: A Creative Approach by Udi Manber (available as a free download)

Preliminaries, continued

- Imagine that our nondeterministic algorithm has as its goal to accept or not accept a language L.

Preliminaries, continued

- Imagine that our nondeterministic algorithm has as its goal to accept or not accept a language L.
- In fact, the above point of view is simply a YES/NO question and is called a decision problem.

Preliminaries, continued

- Imagine that our nondeterministic algorithm has as its goal to accept or not accept a language L.
- In fact, the above point of view is simply a YES/NO question and is called a decision problem.
- Perhaps not surprisingly, *any* algorithm can be viewed as a decision problem (more later).

Preliminaries, continued

- Imagine that our nondeterministic algorithm has as its goal to accept or not accept a language L.
- In fact, the above point of view is simply a YES/NO question and is called a decision problem.
- Perhaps not surprisingly, *any* algorithm can be viewed as a decision problem (more later).
- **Visualization.** Imagine the process of running the nondeterministic algorithm as taking place on a directed graph:

Preliminaries, continued

- Imagine that our nondeterministic algorithm has as its goal to accept or not accept a language L.
- In fact, the above point of view is simply a YES/NO question and is called a decision problem.
- Perhaps not surprisingly, *any* algorithm can be viewed as a decision problem (more later).
- **Visualization.** Imagine the process of running the nondeterministic algorithm as taking place on a directed graph:
- you walk from vertex to vertex along directed edges;

Preliminaries, continued

- Imagine that our nondeterministic algorithm has as its goal to accept or not accept a language L.
- In fact, the above point of view is simply a YES/NO question and is called a decision problem.
- Perhaps not surprisingly, *any* algorithm can be viewed as a decision problem (more later).
- **Visualization.** Imagine the process of running the nondeterministic algorithm as taking place on a directed graph:
- you walk from vertex to vertex along directed edges;
- at some vertices, namely the nd-choice ones, you will have to decide which way to go.

Recognizing a language

- The idea of walking around on a directed graph is this: as you travel, you are on a computation path.

Recognizing a language

- The idea of walking around on a directed graph is this: as you travel, you are on a computation path.
- **Definition.** We say that a nondeterministic algorithm recognizes language L if the following condition is satisfied.

Recognizing a language

- The idea of walking around on a directed graph is this: as you travel, you are on a computation path.
- **Definition.** We say that a nondeterministic algorithm recognizes language L if the following condition is satisfied.
  - Given an input x, it is possible to convert each nd-choice encountered during the execution of the algorithm to a real choice so that

Recognizing a language

- The idea of walking around on a directed graph is this: as you travel, you are on a computation path.
- **Definition.** We say that a nondeterministic algorithm recognizes language L if the following condition is satisfied.
  - Given an input x, it is possible to convert each nd-choice encountered during the execution of the algorithm to a real choice so that
  - the outcome of the algorithm will be to accept x if and only if $x \in L$.

About Decision Problems

- All of the problems we have worked on this semester can be converted to decision problems.

About Decision Problems

- All of the problems we have worked on this semester can be converted to decision problems.

- **Example.** Consider the division algorithm: given nonnegative integers a, b with $b \leq a$, let L be the set of all instances of pairs of non-negative integers (q, r) such that $a = bq + r$ and $0 \leq r < b$.

About Decision Problems

- All of the problems we have worked on this semester can be converted to decision problems.

- **Example.** Consider the division algorithm: given nonnegative integers a, b with $b \leq a$, let L be the set of all instances of pairs of non-negative integers (q, r) such that $a = bq + r$ and $0 \leq r < b$.

- We guess at values of q and r, and if we happen upon a correct guess, the algorithm outputs **ACCEPT**.

About Decision Problems

- All of the problems we have worked on this semester can be converted to decision problems.
- **Example.** Consider the division algorithm: given nonnegative integers a, b with $b \leq a$, let L be the set of all instances of pairs of non-negative integers (q, r) such that $a = bq + r$ and $0 \leq r < b$.
- We guess at values of q and r, and if we happen upon a correct guess, the algorithm outputs **ACCEPT**.
- Otherwise the algorithm outputs **DO NOT ACCEPT**.

Run time and new class of problems

- The run time for a given input $x \in L$ is the length of the shortest computation path that leads to **ACCEPT**.

Run time and new class of problems

- The run time for a given input $x \in L$ is the length of the shortest computation path that leads to **ACCEPT**.
- The run time for the whole nondeterministic algorithm is the worst case run time over all inputs $x \in L$ (ie, everything that leads to **ACCEPT**).

Run time and new class of problems

- The run time for a given input $x \in L$ is the length of the shortest computation path that leads to **ACCEPT**.

- The run time for the whole nondeterministic algorithm is the worst case run time over all inputs $x \in L$ (ie, everything that leads to **ACCEPT**).

- **Definition.** The class of problems for which there exists a nondeterministic algorithm whose run time is polynomial in the size of the input is called **NP**.

Intuition???

- One would think that the nondeterministic polynomial-time algorithms are more powerful than the deterministic polynomial-time algorithms.

Intuition???

- One would think that the nondeterministic polynomial-time algorithms are more powerful than the deterministic polynomial-time algorithms.
- Right?

Intuition???

- One would think that the nondeterministic polynomial-time algorithms are more powerful than the deterministic polynomial-time algorithms.
- Right?
- For example, if we let P be the set of deterministic polynomial-time algorithms, then clearly $P \subset NP$ (why?).

Intuition???

- One would think that the nondeterministic polynomial-time algorithms are more powerful than the deterministic polynomial-time algorithms.
- Right?
- For example, if we let P be the set of deterministic polynomial-time algorithms, then clearly $P \subset NP$ (why?).
- **Challenge.** Is $P = NP$?

Intuition???

- One would think that the nondeterministic polynomial-time algorithms are more powerful than the deterministic polynomial-time algorithms.
- Right?
- For example, if we let P be the set of deterministic polynomial-time algorithms, then clearly $P \subset NP$ (why?).
- **Challenge.** Is $P = NP$?
- Nobody knows!

Skip the final? Get a PhD? Get an A?

- If you show either that $P = NP$ or $P \neq NP$ then you win the following prizes: $\{$A, PhD, skip the final, \$1,000,000 from the Clay Institute[3]$\}$.

[3] http://www.claymath.org/millennium-problems

Skip the final? Get a PhD? Get an A?

- If you show either that $P = NP$ or $P \neq NP$ then you win the following prizes: {A, PhD, skip the final, \$1,000,000 from the Clay Institute[3]}.

- **More Explanation.** A problem X is called NP hard if every problem in the set NP is polynomially reducible to x.

[3] http://www.claymath.org/millennium-problems

Skip the final? Get a PhD? Get an A?

- If you show either that $P = NP$ or $P \neq NP$ then you win the following prizes: {A, PhD, skip the final, \$1,000,000 from the Clay Institute[3]}.
- **More Explanation.** A problem X is called NP hard if every problem in the set NP is polynomially reducible to x.
- A problem X is called NP complete if $X \in NP$ and X is NP hard.

[3] http://www.claymath.org/millennium-problems

Skip the final? Get a PhD? Get an A?

- If you show either that $P = NP$ or $P \neq NP$ then you win the following prizes: {A, PhD, skip the final, \$1,000,000 from the Clay Institute[3]}.
- **More Explanation.** A problem X is called NP hard if every problem in the set NP is polynomially reducible to x.
- A problem X is called NP complete if $X \in NP$ and X is NP hard.
- **Giant Implication:** if any single problem in P is proved to be in NP, then P=NP !!!

[3] http://www.claymath.org/millennium-problems

Skip the final? Get a PhD? Get an A?

- If you show either that $P = NP$ or $P \neq NP$ then you win the following prizes: {A, PhD, skip the final, \$1,000,000 from the Clay Institute[3]}.
- **More Explanation.** A problem X is called NP hard if every problem in the set NP is polynomially reducible to x.
- A problem X is called NP complete if $X \in NP$ and X is NP hard.
- **Giant Implication:** if any single problem in P is proved to be in NP, then P=NP !!!
- **Moral of the story.** Researchers continue collecting NP complete and NP hard problems. The list continues to grow...

[3] `http://www.claymath.org/millennium-problems`

Flashback to 1971 and an example

- Remember Steve Cook and that he started the show with nondeterministic algorithms?

Flashback to 1971 and an example

- Remember Steve Cook and that he started the show with nondeterministic algorithms?
- His theory would have no substance if there were no NP complete problems;

Flashback to 1971 and an example

- Remember Steve Cook and that he started the show with nondeterministic algorithms?
- His theory would have no substance if there were no NP complete problems;
- his great breakthrough came in 1971 when he proved that the **Satisfiability Problem** (or SAT for short) was NP complete and that was the first such problem so proved.

Description of SAT

- Let S be a boolean expression in conjunctive normal form (CNF).

Description of SAT

- Let S be a boolean expression in conjunctive normal form (CNF).
- That is, S is the product (AND) of several sums (OR).

Description of SAT

- Let S be a boolean expression in conjunctive normal form (CNF).
- That is, S is the product (AND) of several sums (OR).
- **Example.** Let $S = (x + y + \overline{z})(\overline{x} + y + z)(\overline{x} + \overline{y} + \overline{z})$,

Description of SAT

- Let S be a boolean expression in conjunctive normal form (CNF).
- That is, S is the product (AND) of several sums (OR).
- **Example.** Let $S = (x + y + \overline{z})(\overline{x} + y + z)(\overline{x} + \overline{y} + \overline{z})$,
- where $\overline{a}$ means "not a" and each expression in parentheses is called a clause.

Description of SAT

- Let S be a boolean expression in conjunctive normal form (CNF).
- That is, S is the product (AND) of several sums (OR).
- **Example.** Let $S = (x + y + \overline{z})(\overline{x} + y + z)(\overline{x} + \overline{y} + \overline{z})$,
- where $\overline{a}$ means "not a" and each expression in parentheses is called a clause.
- The values input for x, y, and z are from the alphabet $\{0, 1\}$, and we do arithmetic $\pmod 2$.

Description of SAT

- Let S be a boolean expression in conjunctive normal form (CNF).
- That is, S is the product (AND) of several sums (OR).
- **Example.** Let $S = (x + y + \overline{z})(\overline{x} + y + z)(\overline{x} + \overline{y} + \overline{z})$,
- where $\overline{a}$ means "not a" and each expression in parentheses is called a clause.
- The values input for x, y, and z are from the alphabet $\{0, 1\}$, and we do arithmetic $\pmod 2$.
- That is, we do arithmetic $\pmod 2$ to determine the value of S, which is, a priori, 0 or 1.

Description of SAT, continued

- A boolean expression S is said to be satisfiable if there is some assignment of 0s and 1s to the variables such that the value of S, upon those inputs, is 1.

Description of SAT, continued

- A boolean expression S is said to be satisfiable if there is some assignment of 0s and 1s to the variables such that the value of S, upon those inputs, is 1.
- **The SAT Problem.** Given a CNF expression S, determine whether or not S is satisfiable (without necessarily finding the satisfying assignment).

Description of SAT, continued

- A boolean expression S is said to be satisfiable if there is some assignment of 0s and 1s to the variables such that the value of S, upon those inputs, is 1.
- **The SAT Problem.** Given a CNF expression S, determine whether or not S is satisfiable (without necessarily finding the satisfying assignment).
- The SAT problem is clearly a decision problem!

Description of SAT, continued

- A boolean expression S is said to be satisfiable if there is some assignment of 0s and 1s to the variables such that the value of S, upon those inputs, is 1.
- **The SAT Problem.** Given a CNF expression S, determine whether or not S is satisfiable (without necessarily finding the satisfying assignment).
- The SAT problem is clearly a decision problem!
- **Example, continued.** The example $S = (x + y + \overline{z})(\overline{x} + y + z)(\overline{x} + \overline{y} + \overline{z})$ is satisfied by $x = 1$, $y = 1$, and $z = 0$ because

Description of SAT, continued

- A boolean expression S is said to be satisfiable if there is some assignment of 0s and 1s to the variables such that the value of S, upon those inputs, is 1.

- **The SAT Problem.** Given a CNF expression S, determine whether or not S is satisfiable (without necessarily finding the satisfying assignment).

- The SAT problem is clearly a decision problem!

- **Example, continued.** The example $S = (x + y + \overline{z})(\overline{x} + y + z)(\overline{x} + \overline{y} + \overline{z})$ is satisfied by $x = 1$, $y = 1$, and $z = 0$ because

- $(1 + 1 + \overline{0})(\overline{1} + 1 + 0)(\overline{1} + \overline{1} + \overline{0}) = 1 \times 1 \times 1 = 1.$

Overview and Summary

- In 1972, Richard Karp proved that 21 important problems[4] were NP complete and the list continues to grow.

[4] https://en.wikipedia.org/wiki/Karp's_21_NP-complete_problems#References

[5] https://www.wolframscience.com/prizes/tm23/turingmachine.html

Overview and Summary

- In 1972, Richard Karp proved that 21 important problems[4] were NP complete and the list continues to grow.

- The idea of the proof that SAT is NP complete is, briefly, that

[4] https://en.wikipedia.org/wiki/Karp's_21_NP-complete_problems#References

[5] https://www.wolframscience.com/prizes/tm23/turingmachine.html

Overview and Summary

- In 1972, Richard Karp proved that 21 important problems[4] were NP complete and the list continues to grow.
- The idea of the proof that SAT is NP complete is, briefly, that
    1. we can guess at a truth assignment and check in polynomial time to see of the outcome of S is 1; if so, then SAT $\in$NP.

[4] https://en.wikipedia.org/wiki/Karp's_21_NP-complete_problems#References

[5] https://www.wolframscience.com/prizes/tm23/turingmachine.html

Overview and Summary

- In 1972, Richard Karp proved that 21 important problems[4] were NP complete and the list continues to grow.
- The idea of the proof that SAT is NP complete is, briefly, that

    1. we can guess at a truth assignment and check in polynomial time to see of the outcome of S is 1; if so, then SAT $\in$NP.
    2. Then it remains to show that SAT is NP hard, the proof of which is nontrivial and uses the idea of a Turing Machine[5].

[4] https://en.wikipedia.org/wiki/Karp's_21_NP-complete_problems#References

[5] https://www.wolframscience.com/prizes/tm23/turingmachine.html

Another example of an NP complete problem

- **Problem:** Given an undirected simple graph G, determine whether or not G can be properly vertex colored with three colors.

Another example of an NP complete problem

- **Problem:** Given an undirected simple graph G, determine whether or not G can be properly vertex colored with three colors.

- **Tool.** 3SAT is NP complete (3SAT is a CNF expression in which all clauses have exactly three variables).

Another example of an NP complete problem

- **Problem:** Given an undirected simple graph G, determine whether or not G can be properly vertex colored with three colors.

- **Tool.** 3SAT is NP complete (3SAT is a CNF expression in which all clauses have exactly three variables).

Another example of an NP complete problem

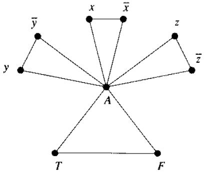
- **Problem:** Given an undirected simple graph G, determine whether or not G can be properly vertex colored with three colors.
- **Tool.** 3SAT is NP complete (3SAT is a CNF expression in which all clauses have exactly three variables).
- **Theorem.** Graph 3-colorability is NP complete.

Another example of an NP complete problem

- **Problem:** Given an undirected simple graph G, determine whether or not G can be properly vertex colored with three colors.
- **Tool.** 3SAT is NP complete (3SAT is a CNF expression in which all clauses have exactly three variables).
- **Theorem.** Graph 3-colorability is NP complete.
- **Idea of Proof.** We will reduce the 3SAT problem to the 3-colorability problem, but first note that if we guess at a 3-coloring of a graph G we can check it's validity in $O(E) = O(V^2)$-time. Why?

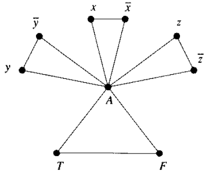
Reduction to 3SAT

- From a CNF expression, we construct a graph as follows. The three colors are T, F, and A.



Reduction to 3SAT

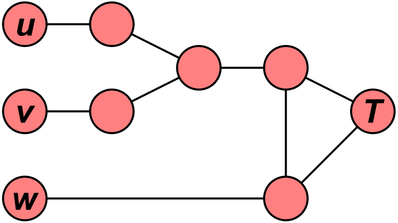
- From a CNF expression, we construct a graph as follows. The three colors are T, F, and A.



- In general, for a CNF expression S that has k variables, there will be $k+1$ triangles at this stage.

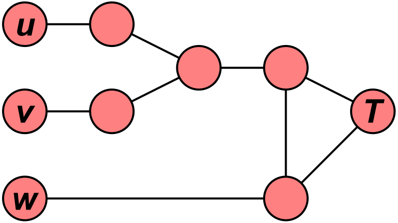
Attach a widget to each clause

- Next, attach the following widget to each clause $U + V + W$ in expression S.



Attach a widget to each clause

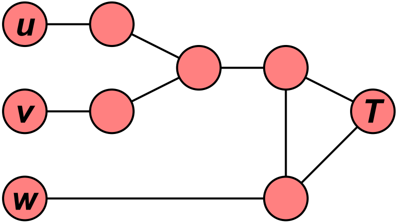
- Next, attach the following widget to each clause $U + V + W$ in expression S.



- **Facts.** The widget graph above is locally 3-colorable and forces one of u, v, or w to be assigned T.

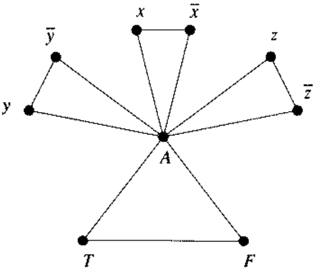
Attach a widget to each clause

- Next, attach the following widget to each clause $U + V + W$ in expression S.



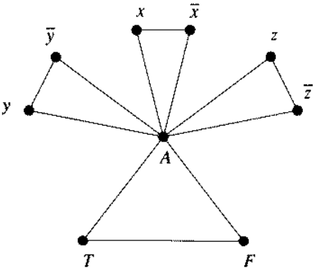
- **Facts.** The widget graph above is locally 3-colorable and forces one of u, v, or w to be assigned T.
- The above fact is important since otherwise S would never be satisfiable.

Widgets and 3-colorability, continued



-

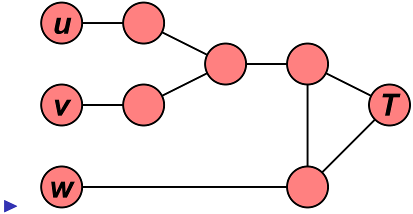
Widgets and 3-colorability, continued



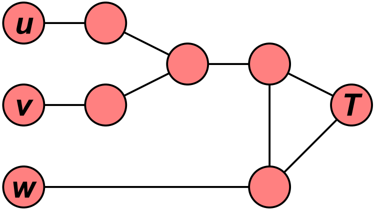
-

- Each of x, $\overline{x}$, y, $\overline{y}$, z, and $\overline{z}$ will never be colored A because all are adjacent to a vertex colored A.

Widgets and 3-colorability, continued

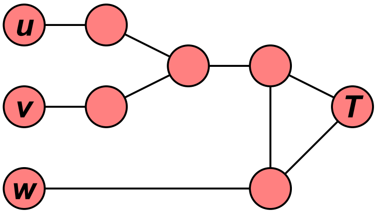


Widgets and 3-colorability, continued

- 

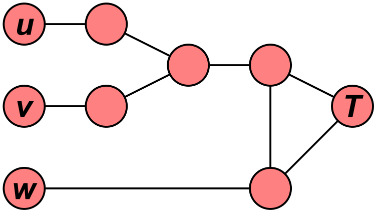
- Thus if x is colored T then $\overline{x}$ must be colored F (since there is always an edge between every pair of variables u and $\overline{u}$).

Widgets and 3-colorability, continued

- 

- Thus if x is colored T then $\overline{x}$ must be colored F (since there is always an edge between every pair of variables u and $\overline{u}$).
- **Conclusion.** We have constructed an infinite family of 3-colorable graphs whose coloring reduces to a known NP-complete problem.

Widgets and 3-colorability, continued

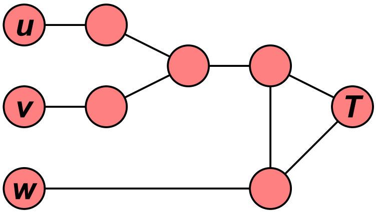
- 

- Thus if x is colored T then $\overline{x}$ must be colored F (since there is always an edge between every pair of variables u and $\overline{u}$).

- **Conclusion.** We have constructed an infinite family of 3-colorable graphs whose coloring reduces to a known NP-complete problem.

- That is, vertex 3-colorability is NP-complete.

Widgets and 3-colorability, continued

- 

- Thus if x is colored T then $\overline{x}$ must be colored F (since there is always an edge between every pair of variables u and $\overline{u}$).
- **Conclusion.** We have constructed an infinite family of 3-colorable graphs whose coloring reduces to a known NP-complete problem.
- That is, vertex 3-colorability is NP-complete.
- **Exercise.** Create the entire graph associated with the example $S = (x + y + \overline{z})(\overline{x} + y + z)(\overline{x} + \overline{y} + \overline{z})$.

Next Week

The Art Gallery Problem

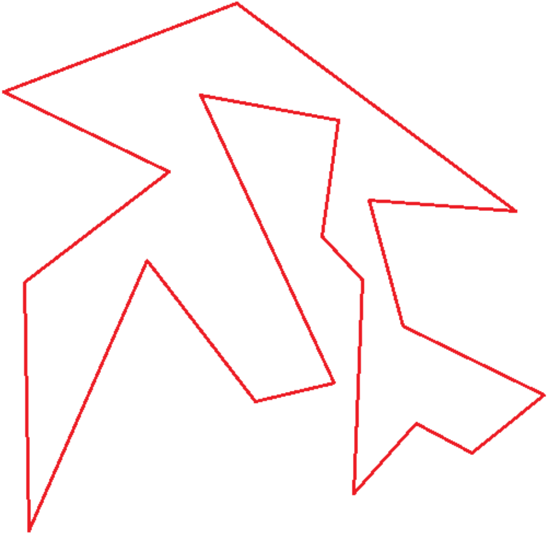


Figure: example caption