Semi-feasible Algorithms
Hem-Ogi Chapter 3

CSC 286/486

Fall 2004

University of Rochester

Anustup Choudhury, Ding Liu, Eric Hughes, Matt Post, Mike Spear, Piotr Faliszewski

Note for digital viewers

- This presentation was put together using Microsoft Powerpoint, from the Office 2004 Suite for the Apple Macintosh. It was then converted to PDF format using the PDF converter built in to Mac OS X's (version 10.3) printing interface
- PDF display was tested under Linux using acroread 5.0.5; xpdf did *not* work

Semi-feasible Algorithms
Hem-Ogi Chapter 3

Section 3.1

Brought to you by:

Anustup Choudhury, Ding Liu, Eric Hughes,
Matt Post, Mike Spear, Piotr Faliszewski

Outline

- Introduction
- Definitions
  - P-sel
  - P/poly
- Tools
  - Tournaments
  - Superloser Theorem
- Main Result
  - P-sel $\subseteq$ P/poly
  - P-sel $\subseteq$ P/quadratic

Semi-feasible Problems

- P – the class of feasible problems
- A nice class, but lacks many interesting languages...perhaps semi-feasible algorithms are also useful?
- Let us consider languages that may not have polynomial-time algorithms, but for which it is possible to efficiently decide which of the two strings given is more likely to be in the language
- One such decision does not give a definite answer as to whether a chosen string is in a language, but perhaps a series of them leads to a solution (of some nice problem)

P-sel • Definition

- Def.

  Set *B* is P-selective if there exists a function *f*,

  $$f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$

  such that:

  - *f* is polynomial-time computable
  - $f(x,y) \in \{x,y\}$
  - $\{x,y\} \cap B \neq \emptyset \Rightarrow f(x,y) \in B$

  In other words: *f* always picks one of its inputs, and if it can pick a one that is in *B* then it does so. "Function *f* picks the input that is no less likely to be in the set."

  Such a function *f* is called a **selector function**

P-sel • Definition (cont'd)

- Def.
  - P-sel is the set of of all P-selective languages
- $P \subseteq$ P-sel
  - The selector function can simply test membership of both of its arguments
  - Let $A \in P$. We define the selector function f to be:
    - $f(x,y) = x$ if $x \in A$
    - $f(x,y) = y$ otherwise
- Is P a proper subset of P-sel?
  - Yes! That is exactly your homework!
  - Don't worry, we will help ☺

P-sel • Example

- So what sets might be in P-sel, but outside of P?
- Consider the language:

$$L_{\Omega} = \{ x \mid x \geq \Omega \}$$

  Where:
  - Ω is some real number
  - the second occurrence of x is treated as a real number whose binary representation is x
- Why is this language P-selective?
  - $f(x,y) = \max(x,y)$
- What is so special about it?

Recall the Halting Problem...

- Def.

  HP = { x | $x \in L(M_x)$ }
- As we all remember, HP is in RE–RECURSIVE
- Can we use that to our advantage?

P-sel • Example (Cont'd)

- Gregory Chaitin from IBM found a really nice Ω…

$$\Omega_{\mathrm{Ch}} = \sum_{p \in \mathrm{HP}} 2^{-p}$$

- Note: the first occurence of *p* is treated as a string, but the second as an integer
- If $L_{\Omega_{\mathrm{Ch}}}$ were recursive then RECURSIVE=RE
  - But you still need to prove that!

P-sel • NP-hard sets in P-sel?

- P-sel contains sets that are not even decidable!
- It stands to reason that there is some NP-complete language that is P-selective!
  - Um... Er... Well not likely, actually...
- Theorem

  If there exists an NP-hard language *A* such that *A* is P-selective then P=NP

  The opposite direction is clearly true. We have P ⊆ P-sel so

  If P=NP then all NP-complete sets are in P-selective. NP-complete sets are NP-hard.

P-sel • Proof of the previous theorem

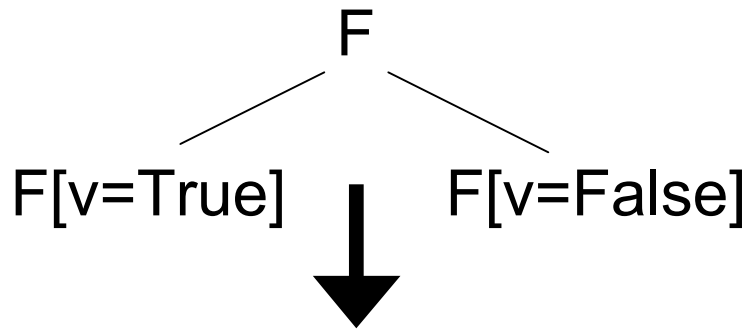
Assumptions:

- *A* – an NP-hard language
- $A \in$ P-sel
- *g* – a selector function for *A*
- *f* – a polynomial-time computable function many-one reducing SAT to *A*

Proof technique:

Show a polynomial-time algorithm for SAT by pruning the tree of possible truth assignments

Let F be the input formula. How do we use *f* and *g* to find a satisfying truth assignment?



g is a selector function—it has to return the string that is no less likely to be in the set.

Since *f* is a reduction, *g* gives us the path in the tree to follow! After *n* steps the ground formula is guaranteed to be satisfiable iff F is satisfiable.

P-sel • Conclusion

- A set is P-selective if given two strings we can decide in polynomial time which of them is no less likely to be in the set
- P-selective sets may be undecidable
- Yet, unless P=NP, none of them can be NP-hard

P/poly • Introduction

- Recall tally languages
  - L is tally if $L \subseteq 1^*$
- What does it take to decide a tally language?
  - L could be undecidable!
  - To decide whether a string of length n is in a tally language you just need to know whether 1^n is in the language...
  - With 1 bit of advice for every length we could decide any tally language...

P/poly • Introduction (Cont'd)

- What about sparse languages?
  - L is sparse if there is a polynomial p such that:
    - $|L^{=n}| \leq p(n)$
- How much advice per length do we need to decide sparse languages?
  - There are at most polynomially many strings of length n
  - Each string is n symbols long
  - If, as a piece of advice, someone gave us all the strings of the given length then we could simply compare each with the input
  - The advice would only be polynomial in size

P/poly • Definition (Informal)

- P/poly is the class of all languages that can be decided given a polynomial amount of advice
- A P/poly **advice interpreter** has to work in polynomal time, but for every string length it is given polynomially many advice bits
- Advice is anything the P/poly algorithm designer wants; it does not even have to be computable…
- ...but it must only depend on the input *length* and not on the input *content*

P/poly • P/poly and sparse sets

- Theorem

  All tally sets are in P/poly

- Theorem

  All sparse languages are in P/poly

- In fact, P/poly is exactly the class of all sets Turing-reducible to sparse sets.
  - Unfortunately, we do not have time to prove this

P/poly • Definition of *A/f*

- What do the symbols mean?
  - A – some language
  - f: N → N – some function
- A/f is the class of all languages L such that for some function h it holds that:
  - $(\forall n)[\ |h(n)| = f(n)\ ]$
  - $L = \{\ x \mid \langle x, h(|x|)\rangle \in A\ \}$
- Intuitions:
  - Function f measures the amount of advice available
  - Language A is the **advice interpreter**
  - Function h provides the advice

P/poly • Definition of *C/F*

- What do the symbols mean?
  - *C* – a class of languages
  - *F* – a set of functions from integers to integers
- *C*/*F* is the class of all languages *L* such that :
  - $(\exists\, A \in C)\ (\exists\, f \in F)\ [\ L \in A/f\ ]$
- Where is P/poly?
  - Take *C* to be the class P, and make *F* the set of all polynomials

P/poly • Example

- Let us formally prove that all tally languages are in P/poly
- Let T be some tally language
- First, select the advice function:
  - We only need 1 bit of advice; $f(n) = 1$
    - $h(n) = 1$ if $1^n \in T$
    - $h(n) = 0$ otherwise
- Advice interpreter:
  - $A = \{\langle x,y \rangle \mid x \in 1^* \text{ and } y=1\}$
- $A \in$ P, f is a polynomial $\Rightarrow$ $T \in$ P/poly

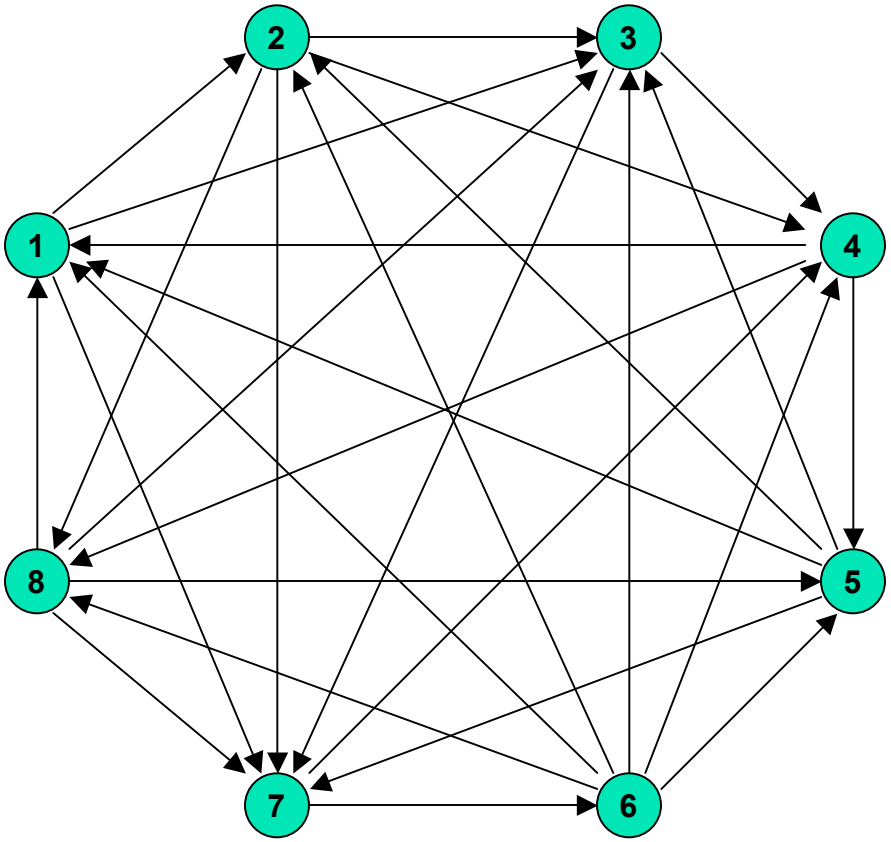
Outline

- Piotr defined for us:
  - P-sel
  - P/poly
- Here's what I'm going to do
  - Explain *k*-tournaments
  - Prove that P-sel $\subseteq$ P/poly

Thm 3.1 • *k*-tournaments

- What is a *k*-tournament?
  - a graph with *k* nodes, and exactly one directed edge between every pair of vertices
- Why is it called a “tournament”?
  - think of each edge as a game played, with the arrow pointing at the winner

Our *k*-tournament (*k*=8)



- arrows point to the winners
- each node is also considered to defeat itself

Thm 3.1 • the superloser set

- Properties of a *k*-tournament
  - *G* = a *k*-tournament graph
    H ⊆ *G*

1) $\|H\| \leq \lfloor \log(k+1) \rfloor$
2) for each $v \in V_G - H$, there is some $g \in H$ such that $(g,v) \in E_G$

  - in other words, there is a subset *H* of *G* whose cardinality is O(log n) in the number of nodes in *G*, and every node in *G* defeats one of the nodes in *H*
  - we call *H* the **superloser** set

Proof 3.1 • process

- At least one person loses half or more of her games (why?)
- Proof procedure: Take that player and remove her from the graph, *G*, as well as everyone who defeated her. Add the player to the loser set, *H*
  - there are between 0 and $\lceil \frac{k}{2} \rceil - 1$ nodes left
- Repeat this process on the remaining graph, *G'*, until there are no nodes left.

Proof of Thm 3.1 (cont'd)

- At each stage we have a full *k*-tournament graph, with *k* shrinking through successive stages
- Eventually there will be no nodes to consider
- The recurrence relation for this is

$$S(0) = 0$$

$$S(k) \leq 1 + S(\lceil \tfrac{k}{2} \rceil - 1) \qquad k \geq 1$$

Review so far

- As Piotr explained:
  - the class P-sel…
    - "semi-feasible" sets: the class of sets that have a selector function that takes two arguments, and returns the one more likely (not less likely) to be in the set
  - the class P/poly
    - sets that can be solved with advice that is the same for all strings *of the same length*
- As I explained
  - *k*-tournaments

Hem-Ogi Thm 3.2 • P-sel ⊆ P/poly

- Proof idea: turn a semi-feasible set into a P/poly set using a *k*-tournament
- Potential point of confusion: the text talks about showing “that semi-feasible sets have small circuits”
  - having small circuits is another characteristic of P/poly; just ignore for now

Proof 3.2 • goal

- Let *L* be our semi-feasible (P-sel) set
- To show that it's in P/poly, we need to show two things

1. $(\forall x)[x \in L \iff \langle x, g(|x|)\rangle \in A]$
   - *A* will be the advice interpreter set from the definition of P/poly
2. $(\exists q)(\forall n)[|g(n)| \leq q(n)]$
   - *q* is the polynomial bound on the advice size

Proof 3.2 • *k*-tournament at $L^{=n}$

- We'll begin by making a *k*-tournament from the elements of $L^{=n}$.
- How?
  - remember that a *k*-tournament is a property of any fully-connected directed graph
  - we can consider each string in $L^{=n}$ to be a node, so all we need is a way to decide which of two strings is the "winner"
- Any ideas?

Proof 3.2 • selector function

- L is a semi-feasible set, so there is a selector function f
- Let $f'(x,y) = f(\min(x,y), \max(x,y))$
  - f' is a selector function for the same language
  - f' is commutative
- The commutativity of f' allows us to construct a k-tournament from the strings in $L^{=n}$
  - call this graph G such that for any $(a,b \in L^{=n} \wedge a \neq b) \Rightarrow ((a,b) \in E_G \Leftrightarrow f'(a,b) = b)$

Proof 3.2 • Properties of $L^{=n}$

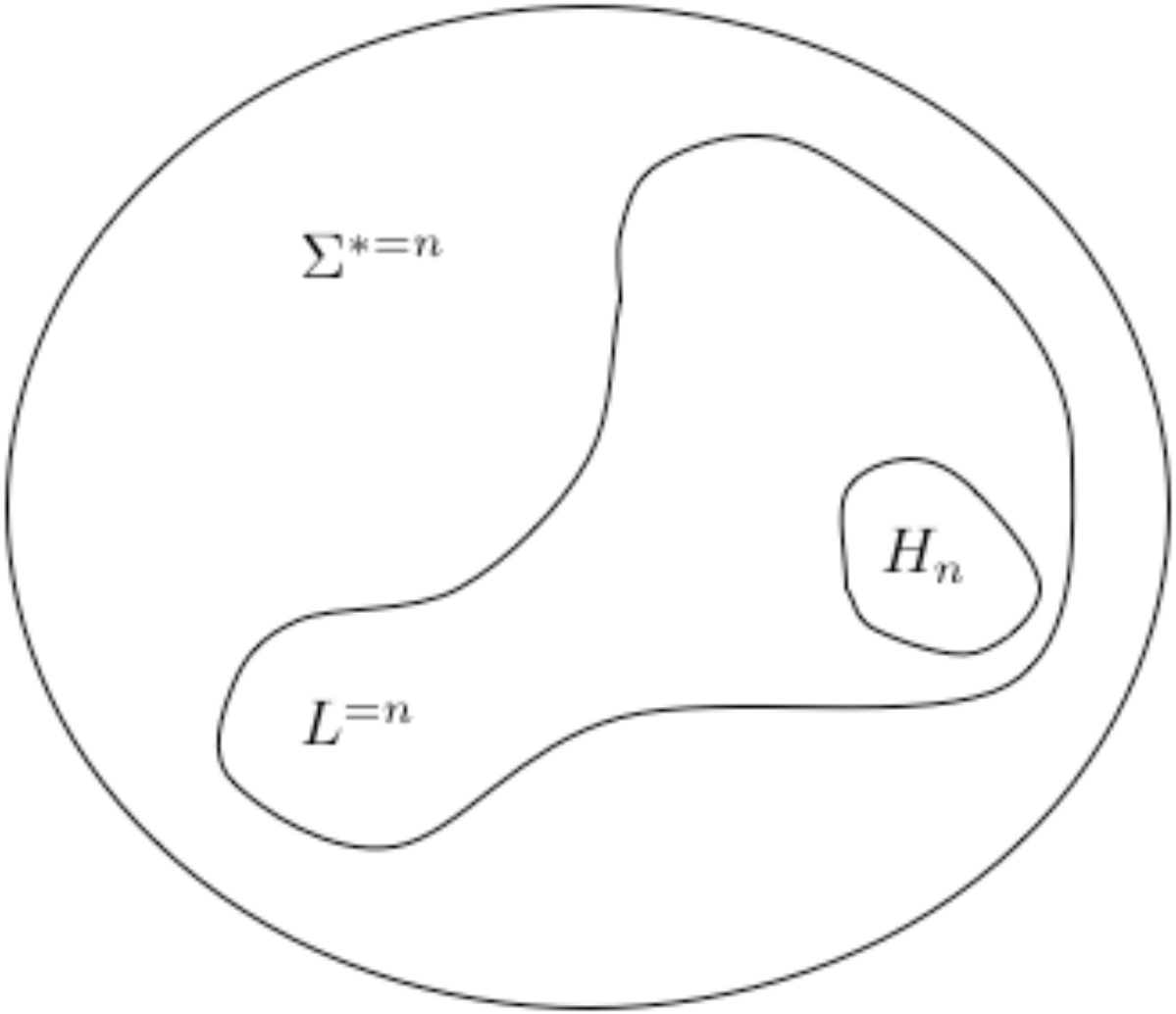
- Because of the *k*-tournament, we know that we have a superloser set $H_n \subseteq L^{=n}$ where:

1) $\|H_n\| \leq \lfloor \log(\|L^{=n}\| + 1) \rfloor$

2) for every element in $L^{=n}$, there exists an $h \in H_n$ such that $f(h,x) = x$

(remember that every superloser beats itself)

Proof 3.2 • map of the world



a map
of the world

Proof 3.2 • what advice?

- So far we have used the selector function to produce a *k*-tournament: Now we'll show our set *L* is in P/poly by providing
  - an **advice function**, *g*, and
  - an **advice interpreter**, the set *A*
- *Remember* that advice on a set is the same for all strings of the same length
- Any guesses what advice *g* gives for *L* on strings of length *n*?

Proof 3.2 • good advice

- The **advice** is H_n: $g(n)$ provides the superloser set at length n
- The **advice interpreter set** is:
  $A = \{\langle x,y\rangle \mid y$ is a (possibly empty) list of elements $v_1, v_2, \ldots, v_z$ and for some j it holds that $f(v_j,x) = x.$
- Clearly, $A \in$ P. Proof:
  On input $\langle x,y\rangle$

  FOR j FROM 1 TO z DO

  IF $f(x,v_j) == x$ ACCEPT

  REJECT

Proof 3.2 • correctness

- Are the requirements met?
  - $A \in P$
  - on the next slide we'll show that *x* is in *L* if and only if $\langle x, g(|x|) \rangle$ is in the advice interpreter set *A* (i.e., that we've met the requirement for a set in P/poly)

Proof 3.2 • verification

- Verification of M_A: three cases
  - $x \in L^{=n} \wedge x \in H_n$
    - *x* is a superloser, so the test for some h_j is whether *f(x,x)* = *x*, which is always true
  - $x \in L^{=n} \wedge x \notin H_n$
    - *x* is not a superloser, but since it is in *L* it will defeat one of the superlosers, so we accept
  - $x \notin L^{=n}$
    - *x* is not a superloser and does not defeat one, so we reject

One for the road

- The class P/poly allows a polynomial number of advice bits. How many did we just use, and what does that do for us?

Semi-feasible Algorithms
Hem-Ogi Chapter 3

Section 3.2

Brought to you by:

Anustup Choudhury, Ding Liu, Eric Hughes,
Matt Post, Mike Spear, Piotr Faliszewski

Outline

- Review
  - P-sel, P/poly
  - P-sel $\subseteq$ P/poly
  - k-Tournaments
- Limited Advice
  - k-Tournament properties
- P-sel $\subseteq$ NP/linear
- Setting the stage for a Polynomial Hierarchy collapse

P-sel

- Set B is P-selective iff $\exists$ function f
  - f is polynomial-time computable
  - $f(x,y) \in \{x,y\}$
  - $\{x,y\} \cap B \neq \emptyset \Rightarrow f(x,y) \in B$
- B has a polynomial-time 2-ary selector function which always chooses the input more likely to be in B

P/poly

- The class of all languages that can be decided in polynomial time with a polynomial amount of advice
  - The advice is polynomial with regard to the length of the string whose membership is being tested
  - Advice is dependent only on input length, not input content
  - The advice doesn't even have to be computable

k-Tournaments

- A graph with k nodes, and exactly one directed edge between every pair of vertices
  - No self loops
- Each arrow represents a game played, with an arrow pointing to the winner
- There is a superloser set of size ≤ $\lfloor \log(k+1) \rfloor$

P-sel ⊆ P/poly

- If we treat the members of our P-sel language as a tournament, then we can use the superloser set as advice
- We can check the membership of an arbitrary string x by applying the selector function on every pair (x,y), where y is a string in the superloser set

Limited Advice

- Did we really only show P-sel ⊆ P/poly?
  - We only used a quadratic amount of advice!
- Can we do better (perhaps by using some nondeterminism?)

The Class PP

- It can be shown that P-sel $\subseteq$ PP/linear
- But PP $\supseteq$ NP

The Class NP

- P/poly ⊆ NP/poly
- P/quadratic ⊆ NP/quadratic
- Can nondeterminism reduce the advice needed to determine membership in polynomial time?

k-Tournaments

Background

- The "l" nodes are a superloser set
- Every node in column x defeats node lx
- There are ⌊log(k+1)⌋ superlosers
- l1 beat l2 (otherwise, l2 would be in l1's column)
- Likewise, l2 beat l3, l3 beat l4 …

Tournament

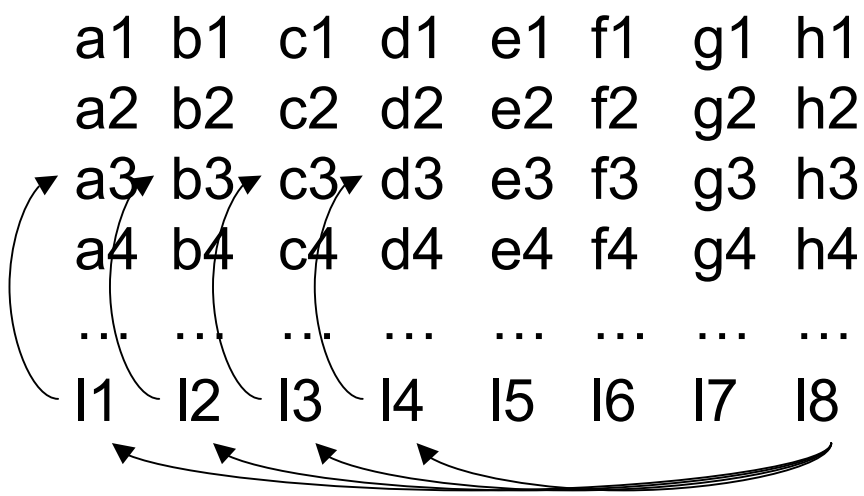
a1	b1	c1	d1	e1	f1	g1	h1
a2	b2	c2	d2	e2	f2	g2	h2
a3	b3	c3	d3	e3	f3	g3	h3
a4	b4	c4	d4	e4	f4	g4	h4
…	…	…	…	…	…	…	…
l1	l2	l3	l4	l5	l6	l7	l8

How much advice is needed?

- No more than the number of superlosers
- When $x \in L$, x defeats at least one superloser
- When $x \notin L$, x does not defeat any superloser

The King Loser

- Look at l8 more closely
- Every superloser beat l8
  - If not, l8 would be in another loser's column
- Everyone else beat a superloser
- Everyone in the tournament is "2 hops" from l8
- l8 is the KING LOSER

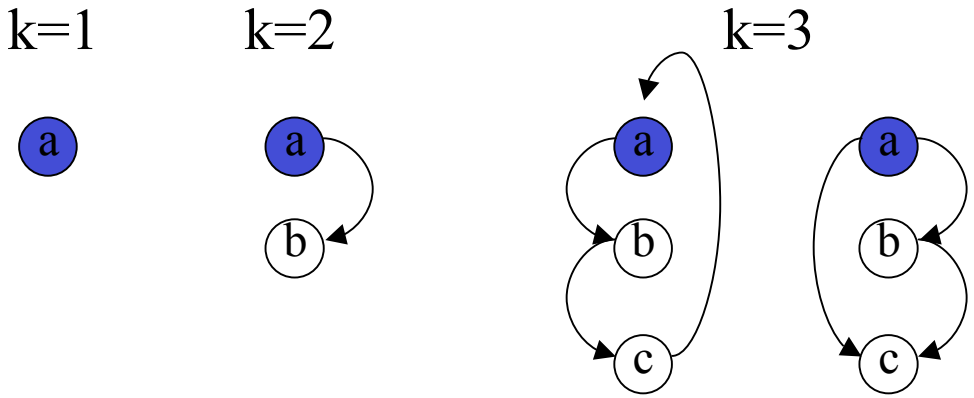


The King Loser Theorem

- If G is a k-tournament, then there is a $v \in V_G$ such that $V_G = R_{2,G}(v)$
- In every k-tournament, there exists a node from which all nodes can be reached via paths of length 2 or less

Proof of the King Loser Theorem

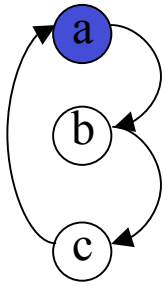
- Proof by induction
- Base case: for k-tournaments whose size ≤ 3, it is obviously true



- Node *a* is always a King Loser
- Some graphs have several King Losers
- Disclaimer: The k=1 base case would suffice for the following proof

Proof of the King Loser Theorem

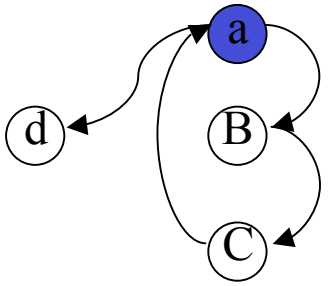
- When k > 3, we use induction
- Recall when k=3



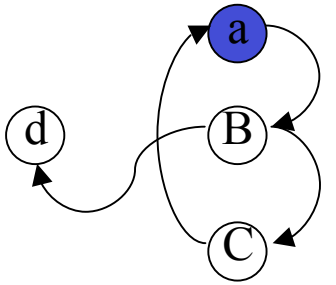
- For any k-tournament, we can classify every node as follows:
  1. x is the king loser
  2. x beat the king loser
  3. x beat someone who beat the king loser
     - (this implies that the king loser beat x)!
- This classification into sets is clear in our k=3 example above
  - (a is in set 1, b is in set 2, c is in set 3)

Proof of the King Loser Theorem

- What happens when we add a new node?
  - Case 1: The new node (d) beats the king loser (a)



  - Case 2: The new node (d) beats someone who beats the king loser (that is, some node in set B)

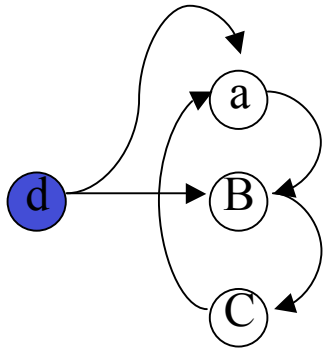


(B and C are sets and may contain multiple nodes

- Either way, the King Loser doesn't change

Proof of the King Loser Theorem

- Case 3: If cases 1 and 2 do not hold, then the new node becomes the king loser
  - The king loser (a) beat the new node (d)
    - otherwise case 1
  - Everyone who beat the king loser (everyone in set B) beat the new node (d)
    - otherwise case 2
  - Everyone else (all nodes in set C) is no farther from the new node (d) than from the old superloser (a)!
    - c beat b, b beat d, and b beat a



(B and C are sets and may contain multiple nodes

Using the King Loser

- Let A ∈ P-sel via commutative selector function f. Consider using f to build a tournament on the nodes in $A^{=n}$ (we'll be vague on uniformity but it isn't a problem here)
- If we knew the King Loser, we could use the following algorithm to determine the membership of x in $A^{=n}$:

```
If x = King Loser, accept
ElseIf f(x, King Loser)=x, accept
Else
   Nondeterministically guess a string
        y of the same length as x
   On each path, if f(x,y)=x and
        f(y,King Loser)=y, accept
Reject
```

Encoding the King Loser

- We want to show that P-sel $\subseteq$ NP/linear
- The King Loser looks like sufficient advice
- How do we encode the King Loser?
  - There is a distinct King Loser for each length n
  - What if $L^{=n} = \varnothing$?

Using n+1 Bits

- Let us define the advice function g(x) as follows:

$$\mathbf{g(x)} = \begin{cases} 1^{n+1} & | \; L^{=n} = \emptyset \\ 0w_n & | \; otherwise \end{cases}$$

- where w_n is the King Loser for strings of length n

Using n+1 Bits

- Let us define the advice interpreter

  $A = \{ \langle x, 0w \rangle \mid$ there is a path of length at most two, in the tournament induced on $L^{=n}$ by f, from w to x$\}$
- We hard-code the case of $\varepsilon \in L$
- Since the selector function f is deterministic and takes polynomial time, we can construct a NPTM to decide A.

Using n+1 Bits

- Recall this algorithm from before

```
If x = King Loser, accept
ElseIf f(x, King Loser)=x, accept
Else
  Nondeterministically guess a string
      y of the same length as x
  On each path, if f(x,y)=x and
      f(y,King Loser)=y, accept
Reject
```

- Since f is a deterministic polynomial-time function, this is clearly a nondeterministic polynomial-time algorithm

Conclusions

- P-sel $\subseteq$ NP/linear
- Since P-sel is closed under complementation
  - P-sel $\subseteq$ coNP/linear
  - P-sel $\subseteq$ NP/linear $\cap$ coNP/linear
- P-sel $\subseteq$ NP/n+1
- Can we do better?
  - NO (see the book for details)

Collapsing the Polynomial Hierarchy

- Using the techniques we've covered so far, we can learn the more subtle properties of the polynomial hierarchy
- But what is the polynomial hierarchy?

$$\mathrm{PH} = \bigcup_i \Sigma_i^p$$

The Polynomial Hierarchy

- A time-bounded analog of the arithmetical hierarchy
- We can think of it iteratively…

$$L \in \Sigma_1^p \Leftrightarrow L = \{x \mid \exists y \in \Sigma^{p'(|x|)} : Q(x,y)\}$$

$$L \in \Sigma_2^p \Leftrightarrow L = \{x \mid \exists y \in \Sigma^{p'(|x|)}, \forall z \in \Sigma^{p''(|x|)} : R(x,y,z)\}$$

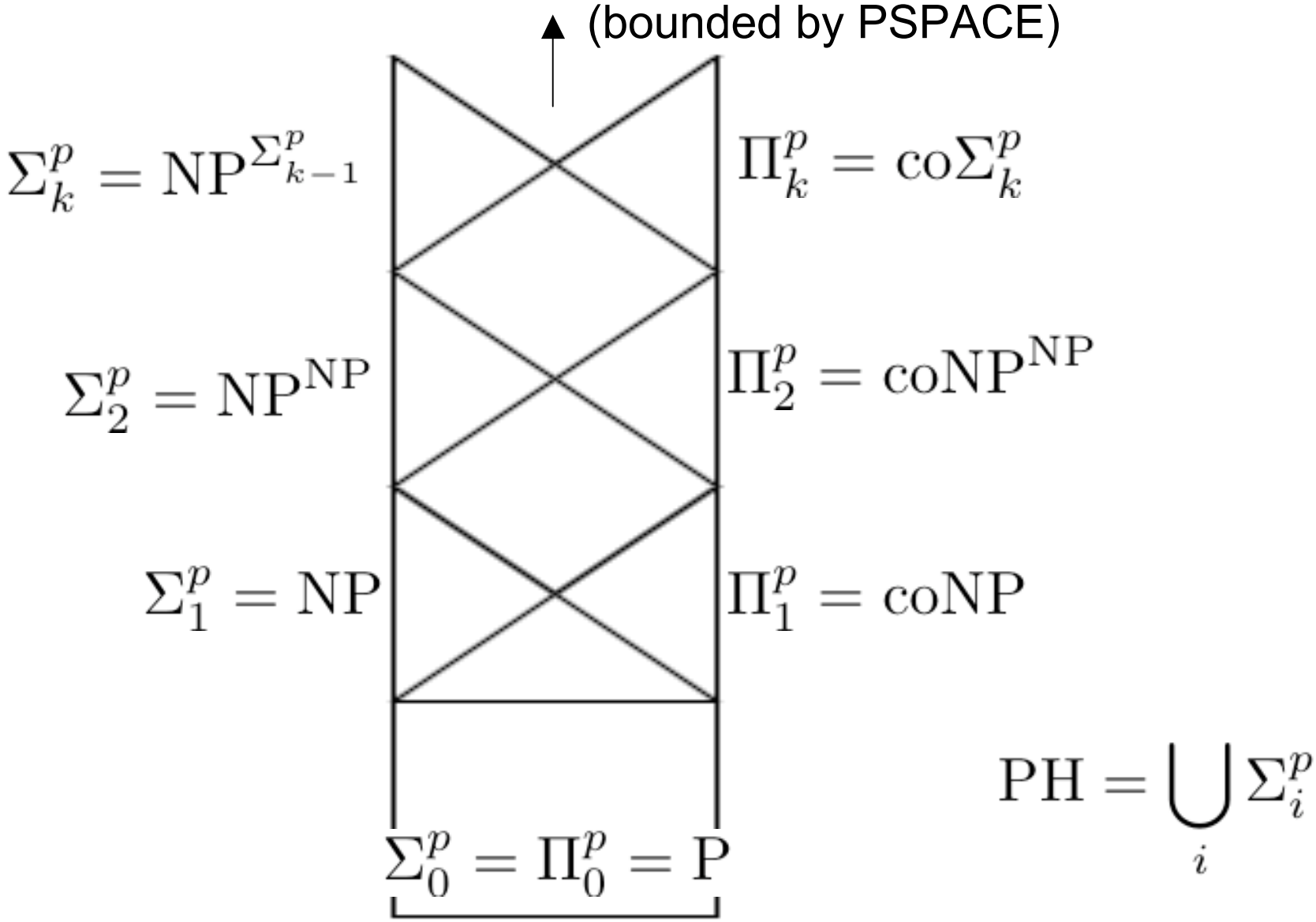
  - Q, R are poly-time predicates; p', p'' are polynomials
  - Π swaps the $\forall$ $\exists$ quantifiers
- … or inductively

$$\Sigma_k^p = \mathrm{NP}^{\Sigma_{k-1}^p}$$

$$\Pi_k^p = \mathrm{co}\Sigma_k^p$$

$$\mathrm{PH} = \bigcup_i \Sigma_i^p$$

The Polynomial Hierarchy



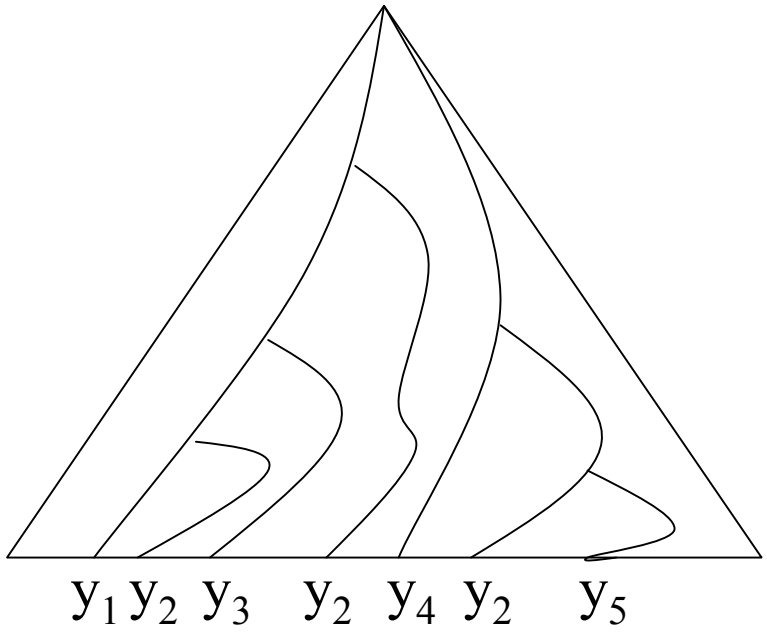
$$\mathrm{PH} = \bigcup_i \Sigma_i^p$$

FP – Deterministic Functions

- A function is in FP iff:
  - It is single valued
  - It is computed by a deterministic, polynomial time TM
- It does not have to be total

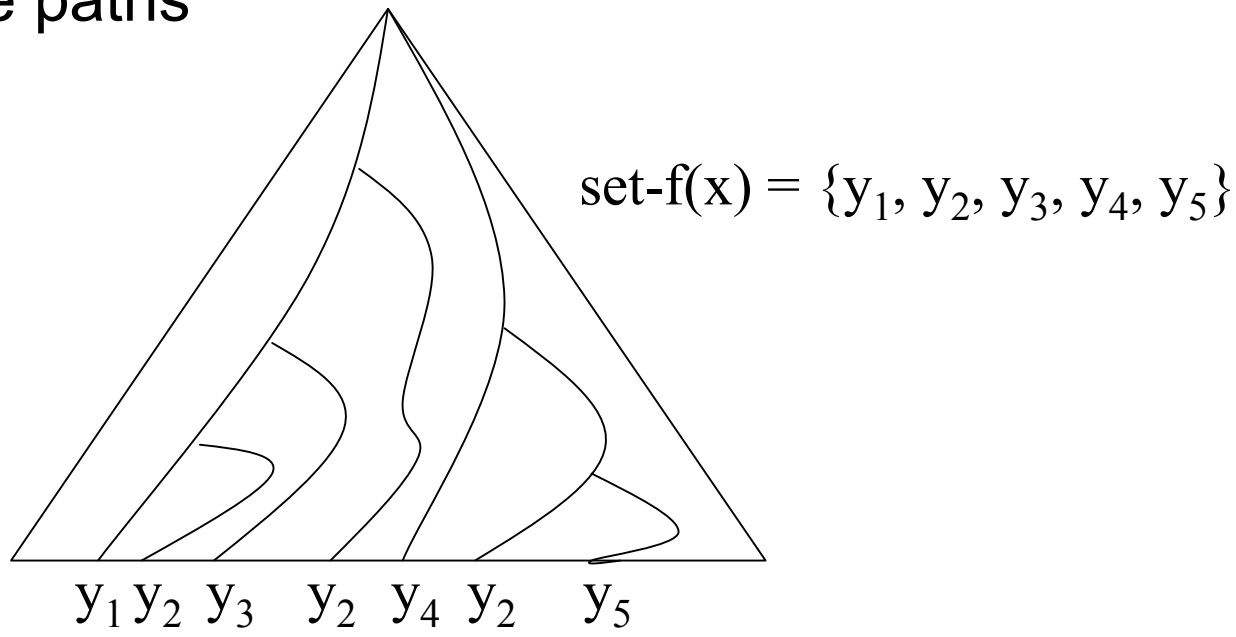
Nondeterministic Functions

- NPMV – nondeterministic, polynomial time, multi-valued
- A function f belongs to NPMV if there exists a NPTM N such that on input x, f's outputs are exactly the outputs of N



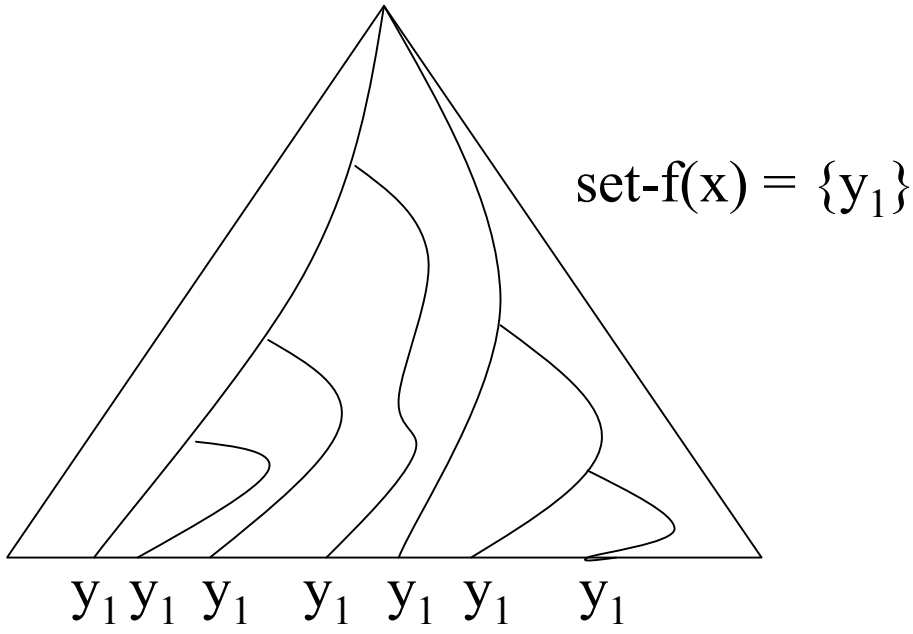
Nondeterministic Functions

- On input x, set-f(x) is the set of all possible outputs of NPMV function f
  - set-f(x) = {a | a is an output of f(x)}
  - On inputs where f(x) is undefined, set-f(x) = ∅
  - We don't care if an item in set-f(x) occurs on multiple paths



Nondeterministic Functions

- NPSV – nondeterministic, polynomial time, single-valued
  - A subset of NPMV where $\forall$ x, ||set-f(x)|| ≤ 1



NPMV Selector Functions

- A set L is NPMV-selective if
  - $\forall$ x,y. set-f(x,y) $\subseteq$ {x,y}
  - $\forall$ x,y. x $\in$ L $\vee$ y $\in$ L $\Rightarrow$ $\emptyset \neq$ set-f(x,y) $\subseteq$ L
- In other words, NPMV-selector functions can return multiple values, but only if both arguments are in L or both arguments are not in L
- The selector function can return $\emptyset$ when both arguments are not in L
- NPSV-selective sets exist as well

Refinement

- NPMV function *f* is a refinement of NPMV function *g* if
  - $\forall$ x. set-f(x) = $\varnothing \Leftrightarrow$ set-g(x) = $\varnothing$
  - $\forall$ x. set-f(x) $\subseteq$ set-g(x)
- A refinement has fewer outputs, but remains defined whenever the original function was defined
- A refinement may be NPSV

Next Time

- Section 3.3: Unique Solutions Collapse the Polynomial Hierarchy

Semi-feasible Algorithms
Hem-Ogi Chapter 3

Section 3.3

Brought to you by:

Anustup Choudhury, Ding Liu, Eric Hughes,
Matt Post, Mike Spear, Piotr Faliszewski

Outline

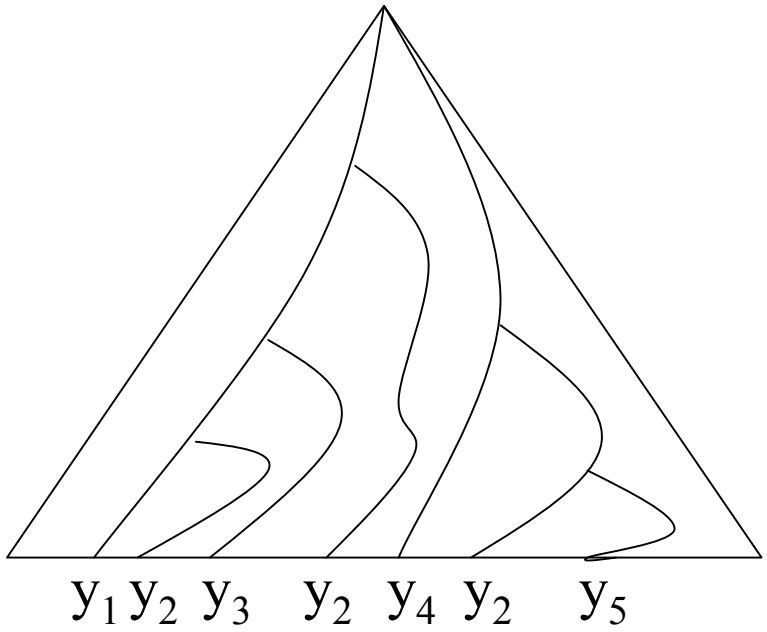
- Review
  - FP, NPMV, NPSV
  - refinements
  - NPMV-sel, NPSV-sel
  - NP/poly, coNP/poly etc.
- Goal
  - If all NPMV functions have NPSV refinements then PH collapses to its second level...
  - ... and even further

FP – Deterministic Functions

- A function is in FP iff:
  - It is single-valued
  - It is computed by a deterministic, polynomial-time TM
- It does not have to be total

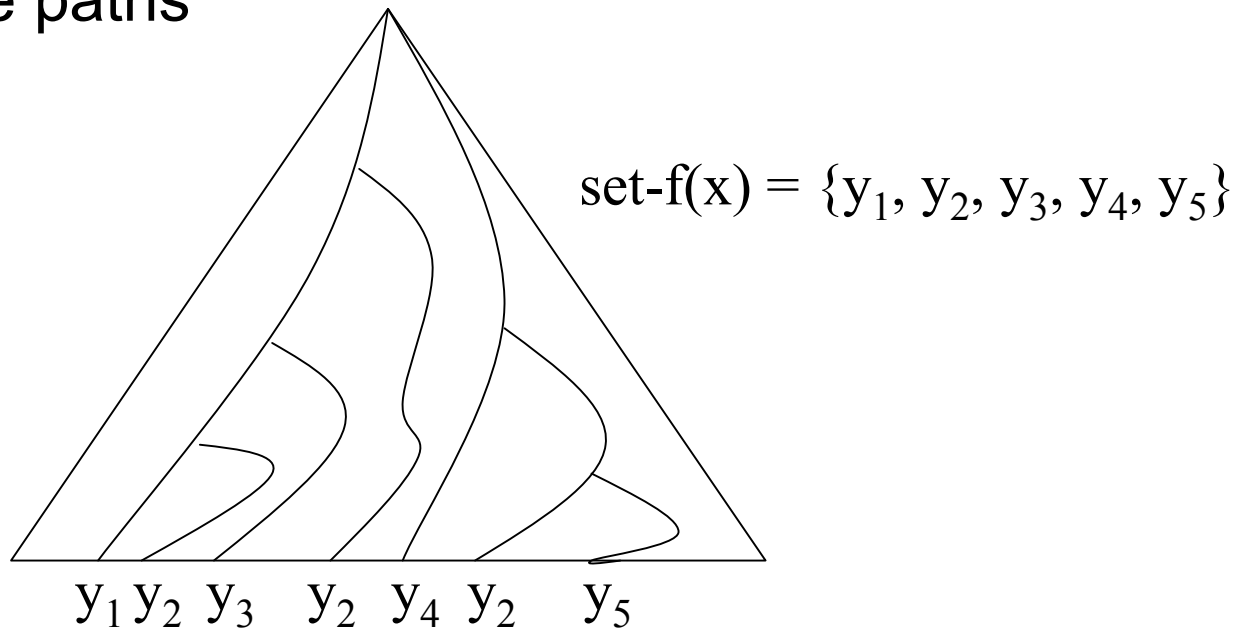
Nondeterministic Functions

- NPMV – nondeterministic, polynomial-time, multivalued
- A function f belongs to NPMV if there exists a NPTM N such that on input x, f's outputs are exactly the outputs of N



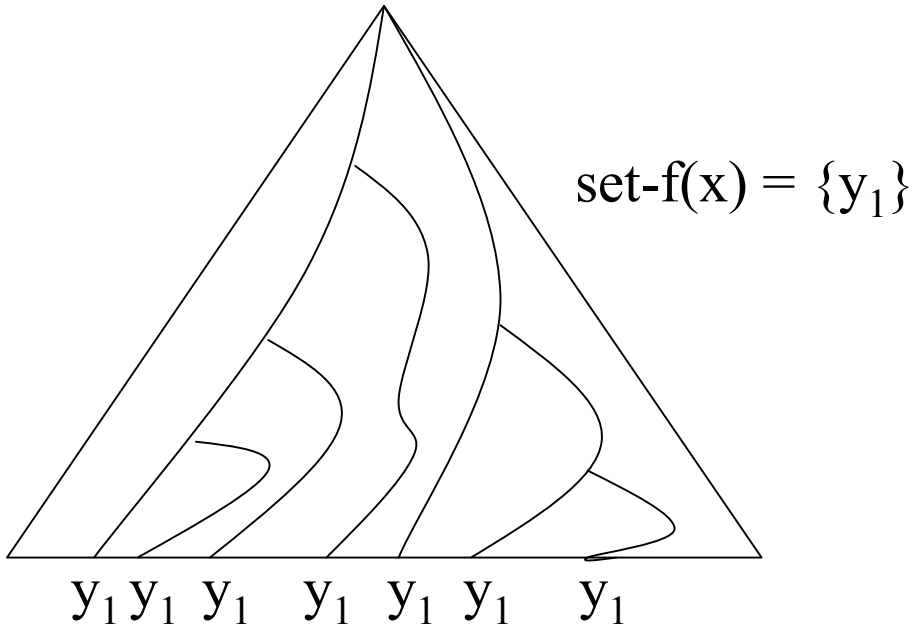
Nondeterministic Functions

- On input x, set-f(x) is the set of all outputs of NPMV function f
  - set-f(x) = {a | a is an output of f(x)}
  - On inputs where f(x) is undefined, set-f(x) = ∅
  - We don't care if an item in set-f(x) occurs on multiple paths



Nondeterministic Functions

- NPSV – nondeterministic, polynomial-time, single-valued
  - A subset of NPMV where $\forall$ x, ||set-f(x)|| ≤ 1



NPMV Selector Functions

- A set L is NPMV-selective if
  - $\forall$ x,y. set-f(x,y) $\subseteq$ {x,y}
  - $\forall$ x,y $(x \in L \vee y \in L) \Rightarrow \emptyset \neq$ set-f(x,y) $\subseteq L$
- In other words, NPMV-selector functions can return multiple values, but only if both arguments are in L or both arguments are not in L
- The selector function can return $\emptyset$ when both arguments are not in L
- NPSV-selective sets exist as well

Refinement

- NPMV function *f* is a refinement of NPMV function *g* if
  - $\forall x.\ \text{set-f}(x) = \varnothing \Leftrightarrow \text{set-g}(x) = \varnothing$
  - $\forall x.\ \text{set-f}(x) \subseteq \text{set-g}(x)$
- A refinement has fewer outputs, but remains defined whenever the original function was defined
- A refinement may be NPSV

Goal

- Theorem

  If all NPMV functions have NPSV refinements then PH collapses to its second level, NP^{NP}.

- We need the following intermediate results:
  - NPSV-sel ∩ NP ⊆ (NP ∩ coNP)/poly
  - NP ⊆ (NP ∩ coNP)/poly ⇒ $PH=NP^{NP}$
- Proof outline
  - Create an NPMV selector for SAT
  - Refine it to be an NPSV selector
  - Conclude NP ⊆ NPSV-sel ∩ NP
  - NP ⊆ NPSV-sel ∩ NP ⊆ (NP ∩ coNP)/poly ⇒ $PH = NP^{NP}$

Goal

- Theorem

  If all NPMV functions have NPSV refinements then PH collapses to its second level, NP^{NP}.

- We need the following intermediate results:
  - NPSV-sel ∩ NP ⊆ (NP ∩ coNP)/poly
  - NP ⊆ (NP ∩ coNP)/poly ⇒ $PH=NP^{NP}$
- Proof outline
  - Create an NPMV selector for SAT
  - Refine it to be an NPSV selector
  - Conclude NP ⊆ NPSV-sel ∩ NP
  - NP ⊆ NPSV-sel ∩ NP ⊆ (NP ∩ coNP)/poly ⇒ $PH = NP^{NP}$

Goal

- Theorem

  If all NPMV functions have NPSV refinements then PH collapses to its second level, NP^{NP}.

- We need the following intermediate results:
  - NPSV-sel ∩ NP ⊆ (NP ∩ coNP)/poly
  - NP ⊆ (NP ∩ coNP)/poly ⇒ $PH=NP^{NP}$
- Proof outline
  - Create an NPMV selector for SAT
  - Refine it to be an NPSV selector
  - Conclude NP ⊆ NPSV-sel ∩ NP
  - NP ⊆ NPSV-sel ∩ NP ⊆ (NP ∩ coNP)/poly ⇒ $PH = NP^{NP}$

Roadmap

- Assume all NPMV functions have NPSV refinements
- Prove: SAT has an NPMV selector function
- If: SAT has an NPSV selector function
- Then: $NP \subseteq NP \cap NPSV\text{-}sel$
- Prove: $NP \cap NPSV\text{-}sel \subseteq (NP \cap coNP)/poly$
- Prove: $NP \subseteq (NP \cap coNP)/poly \Rightarrow PH=NP^{NP}$

NPMV selector for SAT

- NPMV selector for SAT:
  - $f_{SAT}(x,y) = \{x,y\} \cap SAT$
  - Clearly, f_{SAT} is a selector
  - Is it in NPMV?
    - Nondeterministically choose x or y
    - Guess a satisfying truth assignment for the string chosen
    - Check if it indeed is satisfying, and output the chosen string if so
  - SAT $\in$ NPMV-sel

NPSV selector for SAT

- Assumptions
  - All NPMV functions have NPSV refinements
- f_{SAT} is an NPMV selector for SAT
  - We can refine it to be an NPSV selector!

Roadmap

- Assume all NPMV functions have NPSV refinements
- ✓ Prove: SAT has an NPMV selector function
- ✓ If: SAT has an NPSV selector function
- Then: $NP \subseteq NP \cap NPSV\text{-}sel$
- Prove: $NP \cap NPSV\text{-}sel \subseteq (NP \cap coNP)/poly$
- Prove: $NP \subseteq (NP \cap coNP)/poly \Rightarrow PH = NP^{NP}$

NP ⊆ NP ∩ NPSV-sel

- Assumptions
  - All NPMV functions have NPSV refinements
- Under these assumptions SAT is in NPSV-sel (it has an NPSV selector function)
- If NPSV-sel was closed under polynomial-time many-one reductions then we would be done

NPSV-sel

- Theorem

  NPSV-sel is closed under many-one polynomial-time reductions

- Proof
  - A polynomial-time many-one reduces to B
  - B ∈ NPSV-sel
  - f_B – NPSV selector for B
  - g – FP function many-one reducing A to B

$$\text{set}-f_A(x,y) = \begin{cases} \{x\} & \text{if} \quad f_B(g(x),g(y)) = \{g(x)\} \\ \{y\} & \text{if} \quad f_B(g(x),g(y)) = \{g(y)\} \\ \emptyset & \text{if} \quad f_B(g(x),g(y)) = \emptyset \end{cases}$$

Roadmap

- Assume all NPMV functions have NPSV refinements
- ✓ Prove: SAT has an NPMV selector function
- ✓ If: SAT has an NPSV selector function
- ✓ Then: $NP \subseteq NP \cap NPSV\text{-}sel$
- Prove: $NP \cap NPSV\text{-}sel \subseteq (NP \cap coNP)/poly$
- Prove: $NP \subseteq (NP \cap coNP)/poly \Rightarrow PH = NP^{NP}$

NPSV-sel ∩ NP ⊆ (NP ∩ coNP)/poly

- Assumptions
  - L ∈ NPSV-sel ∩ NP
  - N_L – an NPTM accepting L
  - f – an NPSV selector for L
  - set-f(x,y) = set-f(y,x)
- Goal
  - Show that L ∈ (NP ∩ coNP)/poly
  - Proof is essentially the same as in section 3.1, but with a more carefully chosen advice string.
  - We need to provide:
    - An advice interpreter that belongs to NP ∩ coNP
    - Advice of at most polynomial size

NPSV-sel ∩ NP ⊆ (NP ∩ coNP)/poly

- Advice interpreter:
  - Input: <x,d>
  - Interpret d as $<<a_1,\ldots,a_m>,<w_1,\ldots,w_m>>$
    - List of strings $a_1, \ldots, a_m$, each of length |x|
    - List of strings $w_1, \ldots, w_m$
  - Output:
    - Accept if for every i, w_i is an accepting computation path of N_L on a_i, and set-f(x,a_j) = {x} for at least one j
    - Reject otherwise
- Clearly, it is an NP algorithm
- It is also a coNP algorithm
  - Discussed during lecture
- The advice
  - a_i – superloser set for a tournament induced by f on $L^{=n}$
  - w_i – accepting computation paths for a_i's.

Roadmap

- Assume all NPMV functions have NPSV refinements
- ✓ Prove: SAT has an NPMV selector function
- ✓ If: SAT has an NPSV selector function
- ✓ Then: $NP \subseteq NP \cap NPSV\text{-}sel$
- ✓ Prove: $NP \cap NPSV\text{-}sel \subseteq (NP \cap coNP)/poly$
- Prove: $NP \subseteq (NP \cap coNP)/poly \Rightarrow PH=NP^{NP}$

Relativization

- Theorem X

  If $A \in P/poly$ then $\bar{A} \in P/poly$

- Proof
  - Advice interpreter for $\bar{A}$ simulates the advice interpreter for A, and flips its answer
  - Advice is the same
- Relativized version Theorem X

  If $A \in P^B/poly$ then $\bar{A} \in P^B/poly$
  - The same proof works!
  - We say that Theorem X relativizes

Relativization (Cont'd)

- Most of theorems relevant to complexity theory relativize
  - There are some exceptions, though.
  - Nonrelativizing theorems are usually very hard to prove
- A theorem resolving the P vs. NP problem cannot relativize
  - There is a set A such that $P^A = NP^A$
  - There is a set B such that $P^B \neq NP^B$

$NP \subseteq (NP \cap coNP)/poly \Rightarrow PH=NP^{NP}$

- The Karp-Lipton Theorem
  - $NP \subseteq P/poly \Rightarrow PH=NP^{NP}$
  - This theorem relativizes
    - Let A be some language
    - $NP^A \subseteq P^A/poly \Rightarrow PH^A=NP^{NP^A}$
- Assumptions
  - $NP \subseteq (NP \cap coNP)/poly$
  - SAT in $(NP \cap coNP)/poly$ via $NP \cap coNP$ set B
- Proof
  - $NP^B \subseteq P^B/poly \Rightarrow PH^B = NP^{NP^B}$
  - $NP^B = NP$ because $NP^{NP \cap coNP} = NP$, $PH^B = PH$
  - $NP \subseteq P^B/poly \Rightarrow PH=NP^{NP}$
  - NP is a subset of $P^B/poly$ because
    - SAT in $P^B/poly$
    - $P^B/poly$ closed under many-one reductions

Conclusion

- We have reached our goal!
  - We proved all intermediate results
  - It holds that if all NPMV functions have an NPSV refinement then PH collapses to its second level
- Interpretation
  - What does it mean for an NPMV function to have an NPSV refinement?
  - It means that an NPTM can isolate a single solution from possibly exponentially many
  - Isolating a unique solution for every NPMV function collapses the polynomial hierarchy to its second level