Termination
Lecture 3 – Bigger & Bigger

Jonathan Kalechstain
Tel Aviv University

Readings

- Ackermann's Function is Not Primitive Recursive (Chi Woo & Planet Marth)
- Accessible Independence Results for Peano Arithmetic (Laurie Kirby & Jeff Paris)
- The Hydra Battle Revisited (Nachum Dershowitz & Georg Moser)

Overview

- Ackermann’s function is not primitive recursive
- The Ordinals
- Hydra Vs. Hercules : introduction, termination proof, Peano Arithmetic doesn’t suffice

Ackermann's Function

A(m,n) (Ackerman) is defined by recursion:

Base:

1. $A(0,n) = n + 1$
2. $A(m + 1,0) = A(m,1)$

Step:

3. $A(m + 1, n + 1) = A(m, A(m + 1, n))$

Notation:

1) i.h stands for induction hypothesis
2) Ai stands for the i'th rule of ackermann's function
3) APi stands for the I'th ackermann's property proven

Some Ackerman's properties(1):

1. A(m,n) > m +n

Proof by induction on (m,n)

A(m,n) (Ackerman) is defined by recursion:

Base:

1. $A(0,n) = n + 1$

2. $A(m+1,0) = A(m,1)$

Step:

3 . $A(m+1,n+1) = A(m,A(m+1,n))$

1. $A(0,n) =_{A1} n + 1 > n$
2. $A(m+1,0) =_{A2} A(m,1) >_{i.h} m + 1$
3. $A(m+1,n+1) =_{A3} A\big(m,A(m+1,n)\big) >_{i.h} m$ $+ A(m+1,n) \geq_{i.h} m + n + 2$

Some Ackerman's properties(2):

2. $x > y \rightarrow A(m,x) > A(m,y)$
Proof by induction on (m,x)

A(m,n) (Ackerman) is defined by recursion:
Base:
1. $A(0,n) = n+1$
2. $A(m+1,0) = A(m,1)$
Step:
3 . $A(m+1,n+1) = A(m, A(m+1,n))$

1. $A(0,x) =_{A1} x+1 > y+1_{A1} = A(0,y)$
2. $A(m+1,x+1) =_{A3} A\big(m, A(m+1,x)\big) >_{i.h} A\big(m, A(m+1,y)\big) =_{A3} A(m+1,y+1)$

Some Ackerman's properties(3):

3. $x > y \rightarrow A(x,n) > A(y,n)$

Proof by induction on (x,n)

A(m,n) (Ackerman) is defined by recursion:

Base:

1. $A(0,n) = n + 1$
2. $A(m+1,0) = A(m,1)$

Step:

3 . $A(m+1,n+1) = A(m, A(m+1,n))$

1. $A(x,n) >_{AP1} x + n \geq n + 1 = A(0,n)$
2. $A(x+1,0) =_{A2} A(x,1) >_{i.h} A(y,1) =_{A2} A(y+1,0)$
3. $A(x+1,n+1) =_{A3} A\big(x, A(x+1,n)\big) >_{i.h} A\big(y, A(x+1,n)\big)$
   $>_{i.h+AP2} A\big(y, A(y+1,n)\big) = A(y+1,n+1)$

Some Ackerman's properties(4):

4. $A(m+n+2,x) > A(m,A(n,x))$

Proof by induction on (m+n,x)

A(m,n) (Ackerman) is defined by recursion:

Base:

1. $A(0,n) = n+1$
2. $A(m+1,0) = A(m,1)$

Step:

3 . $A(m+1,n+1) = A(m,A(m+1,n))$

1. $A(n+2,x) >_{AP3} A(n+1,x) \geq_{AP3} A(n,x)+1$
   $=_{A1} A(0,A(n,x))$
2. $A(m+n+2,0) =_{A2} A(m+n+1,1)$
   $>_{i.h} A\big(m,A(n-1,1)\big) =_{A2} A(m,A(n,0))$
3. $A(m+n+2,x+1) =_{A3} A\big(m+n+1,A(n+m+2,x)\big)$
   $>_{i.h+AP2,3} A\big(m,A(n,A(m,x))\big) >_{AP1,2} A\big(m,A(n,x+m)\big)$
   $\geq_{AP2} A(m,A(n,x+1))$

Primitive Recursive Functions(1)

We define the set of all primitive recursive functions $f: N^k \to N$ (when $1 \leq k$) as follow:

1. $f \equiv 0$ is primitive recursive
2. $S(n) = n + 1$ is primitive recursive
3. $f_i^n = (x_1, x_2, \dots, x_i, \dots, x_n) = x_i$ is primitive recursive (projection function)
4. Given $f: N^k \to N$ primitive recursive function, and given $g_1, g_2, \dots g_k: N^m \to N$ primitive recursive functions, the composition $h(x_1, \dots, x_m) = f(g_1(x_1, \dots, x_m), \dots, g_k(x_1, \dots, x_m))$ is a primitive recursive function

Primitive Recursive Functions(2)

5. let $f: N^k \rightarrow N, g: N^{k+2} \rightarrow N$ be primitive recursive functions. The function $h: N^{k+1} \rightarrow N$:

   $h(0, x_1, \dots, x_k) = f(x_1, \dots, x_k)$

   $h(y+1, x_1, \dots, x_k) = g(y, h(y, x_1, \dots, x_k), x_1, \dots, x_k)$

   is primitive recursive.

Example: add(x,y) = x + y is primitive recursive

- Denote $P_i^j(x_1, x_2, \dots, x_i, \dots, x_j) = x_i$
- $add(0, x) = P_1^1(x) = x$
- add(n+1,x)= $S(P_2^3(n, add(n, x), x))$

Note:
in this example:
1. $k = 1$
2. $add = h$
3. $P_1^1 = f: primitive\ recursive\ by\ definition$
4. $g = S(P_2^3): primitive\ recursive\ by\ composition$
   $of\ primitive\ recursive\ functions$

Reminder:

let $f: N^k \to N, g: N^{k+2} \to N$ be primitive recursive functions the function $h: N^{k+1} \to N$:

$$h(0, x_1, \dots, x_k) = f(x_1, \dots, x_k)$$
$$h(y+1, x_1, \dots, x_k)$$
$$= g(y, h(y, x_1, \dots, x_k), x_1, \dots, x_k)$$

is primitive recursive.

Ackermann function is not primitive recursive(1)

Notation:

1. A-Ackerman's function
2. $PR = \{f | f \text{ is primitive recursive}\}$
3. $\vec{x} = (x_1, x_2, \ldots, x_n)$ is a vector of n non negative integers
4. $x = \max\{x_1, x_2, \ldots, x_n\}$
5. $S(n) = n + 1$
6. $z(n) = 0$
7. $P_i^j(x_1, x_2, \ldots, x_i, \ldots, x_j) = x_i$
8. AP-i: the I'th Ackerman property proven in previous slides

Ackermann function is not primitive recursive(2)

Definition :

We say that a function g is majorized by A, if for some $i \in N$, $g(\vec{x}) < A(i, x), \forall \vec{x} \in N^n$

We define:

$A_m = \{g | g \text{ is majorized by } A\}$

Ackermann function is not primitive recursive(3)

- The idea in this proof is to show a quality that holds to all primitive recursive functions, but doesn’t hold for A.
- We show that A “grows” faster than any function in PR.

Ackermann function is not primitive recursive(4)

Proposition 1: $PR \subseteq A_m$

Proof: By structural induction:

Base: $\{S(n), z(n), P_i^j(\vec{x})\} \subseteq A_m$

1. $z(n) = 0 < n + 1 = A(0, n)$ (i=0)
2. $s(n) = n + 1 <_{AP1} A(1, n)$ (i=1)
3. $P_i^j(\vec{x}) = x_i \leq x < x + 1 = A(0, x)$ (i=0)

Reminder:

We say that a function g is majorized by A, if for some i, $g(\vec{x}) < A(i, x), \forall \vec{x} \in N^n$

Ackermann function is not primitive recursive(5)

Closure under composition:

4. Given $f: N^k \to N$ and given $g_1, g_2, \dots g_k: N^m \to N$ functions in A_m:

   we show the composition $h = f(g_1, \dots, g_k)$ is in A_m.

   By the given we know that

   $(*)$ $g_i(\vec{x}) < A(r_i, x)$ and f$(\vec{y}) < A(s, y)$.

   Define: $g_j(\vec{x}) = \max\{g_i(\vec{x})\}_{i \in \{1 \dots k\}}$

   Now: $h(\vec{x}) = f(g_1, \dots, g_k) <_{(*)} A\left(s, g_j(\vec{x})\right)$

   $<_{*+AP2} A\left(s, A(r_j, x)\right) <_{AP4} A(s + r_j + 2, x)$

Reminder:

We say that a function g is majorized by A, if for some i, $g(\vec{x}) < A(i, x), \forall \vec{x} \in N^n$

Ackermann function is not primitive recursive(6)

Closure under primitive recursion:

5. let $f: N^k \to N, g: N^{k+2} \to N$ be in A_m. We show that the function $h: N^{k+1} \to N$:

$h(0, x_1, \dots, x_k) = f(x_1, \dots, x_k)$

$h(y+1, x_1, \dots, x_k) = g(y, h(y, x_1, \dots, x_k), x_1, \dots, x_k)$ is in A_m.

By the given we know that:

$f(\vec{x}) < A(r, x)$ and $g(\vec{y}) < A(s, y)$.

Ackermann function is not primitive recursive(7)

Closure under primitive recursion:

We first prove: $h(n, \vec{x}) < A(q, n + x)$ for some q not depending on x,n.

Let us pick $q = 1 + \max\{r, s\}$ and prove the claim by induction on n:

Base:

n=0:
$h(0, \vec{x}) = f(\vec{x}) < A(r, x) <_{AP3} A(q, x)$

Ackermann function is not primitive recursive(8)

Closure under primitive recursion:

Step:

Assume that $h(n, \vec{x}) < A(q, n + x)$:

$h(n+1, \vec{x}) = g(n, h(n, \vec{x}), \vec{x}) < \mathrm{A(s, z)}$ when $z = \max\{n, h(n, \vec{x}), x\}$.

Now:

1. $\max\{x, n\} \leq n + x <_{AP1} A(q, n + x)$
2. $h(n, \vec{x}) < A(q, n + x)$ (i.h)

We can conclude that: $z < A(q, n + x)$

Ackermann function is not primitive recursive(9)

Closure under primitive recursion:

Step:

Now:

$h(n+1,\vec{x}) < A(s,z) <_{AP2} A\big(s, A(q, n+x)\big)$
$\leq_{AP3} A\big(q-1, A(q, n+x)\big) =_{A-Definition}$
$A(q, n+1+x)$.

Ackermann function is not primitive recursive(10)

Proof that $A(2, z) = 2z + 3$: by induction on z:

Base: A(2,0)=A(1,1)=A(0,A(1,0))=A(1,0)+1=A(0,1)+1=2+1=3

Step: A(2,z+1)=A(1,A(2,z))=A(1,2z+3)=

Auxilary claim : A(1,x)=x+2 (by induction on x)

Base: A(1,0)=A(0,1)=2

Step: A(1,x+1)=A(0,A(1,x))=1+A(1,x)=x+3

Back to proof: A(1,2z+3)=2z+5

Ackermann function is not primitive recursive(11)

Closure under primitive recursion:

Now, let $z = \max\{x, y\}$. By the last claim proven:

$h(y, \vec{x}) < A(q, x + y) \leq_{AP-2} A(q, 2z)$
$<_{AP-2} A(q, 2z + 3) \quad =_{last\ Page\ proof} A\big(q, A(2, z)\big) <_{AP4}$
$A(q + 4, z)$

Proving that $h \in A_m$.

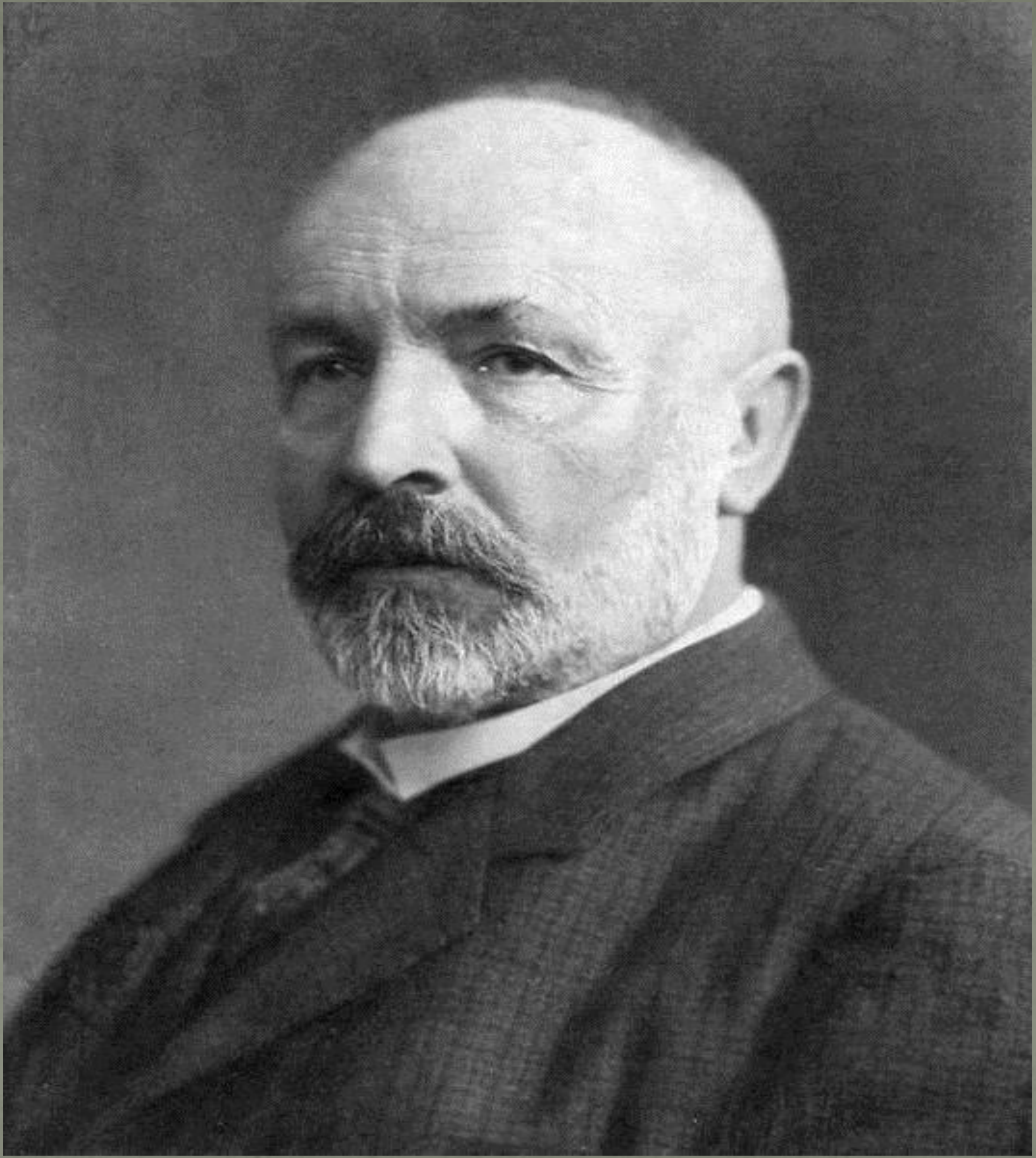
We had just proved that $PR \subseteq A_m$, and because it can't be that $A \in A_m$, we conclude that A is not primitive recursive ■

The Natural Numbers

- Defined as $N = \{0,1,2,3 \ldots.\}$ and is used for two purposes:

1. Count discrete objects (7 pens, 12 children)
2. Orders objects (the seventh child, the fourth pen)

- There is an infinite number of natural numbers ,how can we represent objects in an infinite position ?



- Georg Cantor (1883) invented the notion of ordinals:

- The first ordinals (finite ordinals) are the natural number: 0,1,2,3,4…
- The first infinite ordinal, denoted by ω, is larger than any natural number, but smaller than any other infinite ordinal
- After ω comes $\omega+1$, $\omega+2$, $\omega+3$, … $\omega 2$, $\omega 2+1$,… $\omega 3$,… ω^2, ω^2+1,… ω^3,…, ω^ω,… ω^{ω^ω},….ε_0…

More ordinals (1)

- Ordinals are defined to be the set of all smaller ordinals.
- For any two ordinals α, β we define the following (good) ordering :$\alpha < \beta \leftrightarrow \alpha \in \beta$
- The following ordinals are presented in increasing order, when n is a natural number, and α is an infinite ordinal

- $0 = \emptyset$
- $1 = \{0\} = \{\emptyset\}$
- $2 = \{0,1\} = \{\emptyset, \{\emptyset\}\}$
- $n = \{0,1,2,3 \ldots n-1\}$
- $\omega = \{0,1,2 \ldots\} = N$
- $\omega + 1 = \omega \cup \{\omega\} = \{0,1,2 \ldots \omega\}$
- $\omega + 2 = (\omega + 1) \cup \{(\omega + 1)\} = \{0,1 \ldots, \omega, \omega+1\}$
- $\omega + n = (\omega + (n-1)) \cup \{(\omega + (n-1)\}$
- $\alpha + 1 = \alpha \cup \{\alpha\}$

More ordinals (2)

- $\omega 2 = \omega \cup \{\omega + n | n < \omega\} = \{1,2 \ldots \omega, \omega + 1, \omega + 2 \ldots\}$
- $\omega n = \bigcup_{i<\omega}(\omega(n-1)+i)$
- $\omega^2 = \bigcup_{i<\omega}(\omega i)$
- $\omega^n = \bigcup_{i<\omega}(\omega^{n-1} i)$
- $\omega^\omega = \bigcup_{n<\omega} \omega^n$
- $\omega^\alpha = \bigcup_{\beta<\alpha}(\omega^\beta)$
- $\epsilon_0 = \omega^{\epsilon_0} = \bigcup_{n<\omega} \left.\omega^{\omega^{\cdot^{\cdot^{\omega}}}}\right\} n \text{ } times$
- And more…

Ordinals

$0 < 1 < 2 < \cdots$

$< \omega < \omega+1 < \omega+2 < \cdots$

$< \omega 2 < \omega 2+1 < \cdots < \omega 3 < \cdots < \omega 4 < \cdots$

$< \omega^2 < \omega^2+1 < \cdots < \omega^2+\omega < \omega^2+\omega+1 < \cdots$

$< \omega^3 < \omega^3+1 < \cdots < \omega^4 < \cdots < \omega^5 < \cdots$

$< \omega^\omega < \cdots < \omega^{\omega^\omega} < \cdots < \omega^{\omega^{\omega^\omega}} < \cdots$

Hercules Vs Hydra

The Hydra Battle

- Hercules always wins
- Kirby and Paris prove that more than a regular induction on the natural numbers is needed to prove Hercules's victory.

The Formal Battle (1)

Representation:

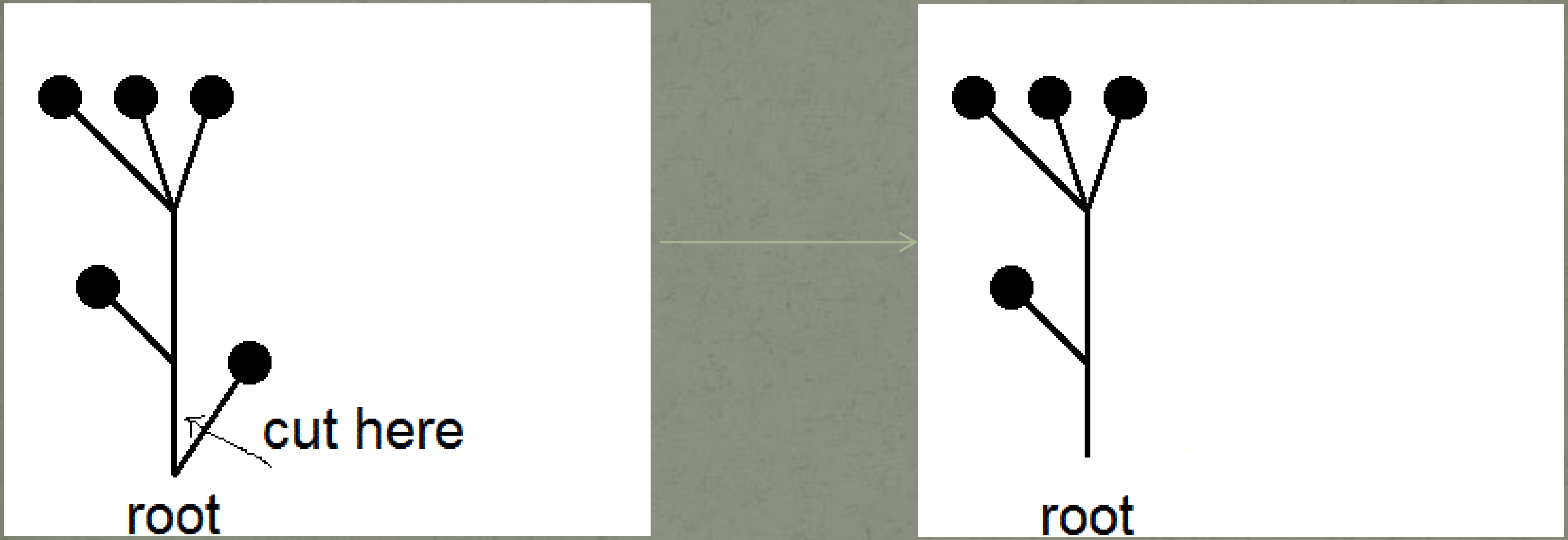
1. Hydras are represented as unordered, rooted, finite tress.
2. Each leaf in the tree represents a head of the hydra.
3. (H,n) describes a single configuration in the game, where H denotes Hydra and n denotes the current stage of the game

What about the rules of the battle?

The Formal Battle(2)

1. At each stage Hercules can decide which head to decapitate.
2. If the head (leaf) decapitated has no grandparent node, then hydra suffers lost.
3. If the head (leaf) does have a grandparent node, then Hydra multiply the branch issuing from that node along with the damaged subtree by a factor n (the stage).
4. Hercules wins when hydra is reduced to an empty tree

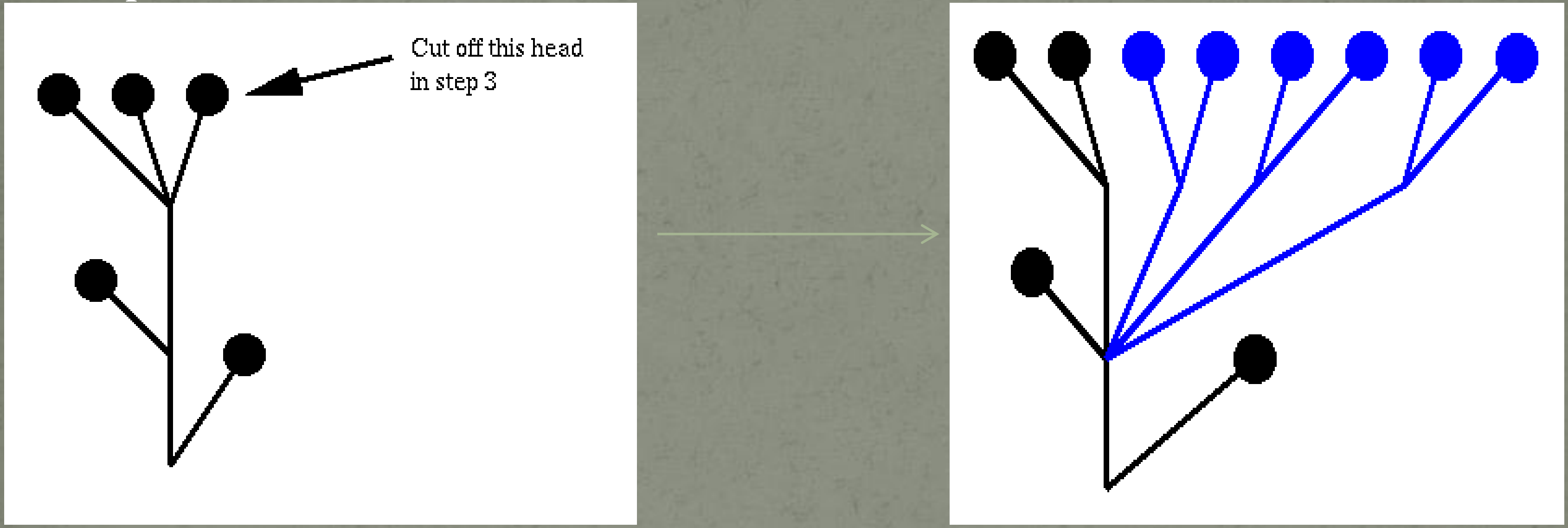
Example (1)



Hydra suffers lost as node had no grandparent root

Example(2)

Delete the head and its neck. Descend down by 1 from the node at which the neck was attached. Look at the subtree growing from the connection through which you just descended. Grow 3 copies of that subtree

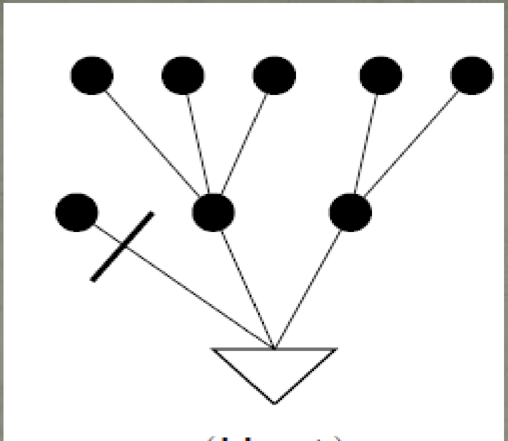


The node did have a grandparent, at stage n=3!

Functional Hydra

- We deal with ordered trees (with immediate subtrees ordered sequentially from left to right)
- Nil denote an empty list.
- cons(x,y) is the list obtained by prepending an element x (the right-most subtree) to a list y (the remaining tree)
- car(cons(x,y))=x
- cdr(cons(x,y))=y

Example


$(\mathsf{H}_1, 1)$

This example can be written as

$$\mathtt{cons}(\ell, \mathtt{cons}(\mathtt{cons}(\mathtt{cons}(\ell, \mathtt{cons}(\ell, \mathtt{nil})), \mathtt{nil}), \mathtt{cons}(\ell, \mathtt{cons}(\ell, \mathtt{cons}(\ell, \mathtt{nil}))))) \ ,$$

When we write l=cons(nil,nil) for a leaf and we have reordered the immediate subtrees of the root of H_1 such that the number of nodes is non-increasing

Encoding the hydra battle(1):

$$h_0(x)$$

where

$$h_n(x) \quad := \begin{cases} \mathtt{nil} & x = \mathtt{nil} \\ h_{n+1}(d_n(x)) & \text{otherwise} \end{cases}$$

$$d_n(x) \quad := \begin{cases} \mathtt{cdr}(x) & \mathtt{car}(x) = \mathtt{nil} \\ f_n(\mathtt{cdr}(\mathtt{car}(x)), \mathtt{cdr}(x)) & \mathtt{car}(\mathtt{car}(x)) = \mathtt{nil} \\ \mathtt{cons}(d_n(\mathtt{car}(x)), \mathtt{cdr}(x)) & \text{otherwise} \end{cases}$$

$$f_n(y, x) := \begin{cases} x & n = 0 \\ \mathtt{cons}(y, f_{n-1}(y, x)) & \text{otherwise} \end{cases}$$

Encoding the hydra battle(2):

- $H_n(x)$ plays the game with initial hydra x beginning at stage n.
- $d_n(x)$ plays one round of the battle, travelling left along a leftmost branch until encountering a branch z such that $car(car(z))$ is empty, using $f_n(y, x)$ to prepend k copies of $y = cdr(car(z))$ to x.

Well founded Ordering(1)

- A partial order $>$ is an irreflexive and transitive binary relation (for example the regular > on the Naturals)
- A partial order $>$ on a set A is well founded if there exists no infinite descending sequence $a_1 > a_2 > \cdots$ of elements from A (for example (N, $>$)) is a well founded ordering)
- A partial order $>$ is linear if $\forall a, b \in A, s.t\ a \neq b$ a and b are comparable by $>$.
- A linear well founded order is called a well-order

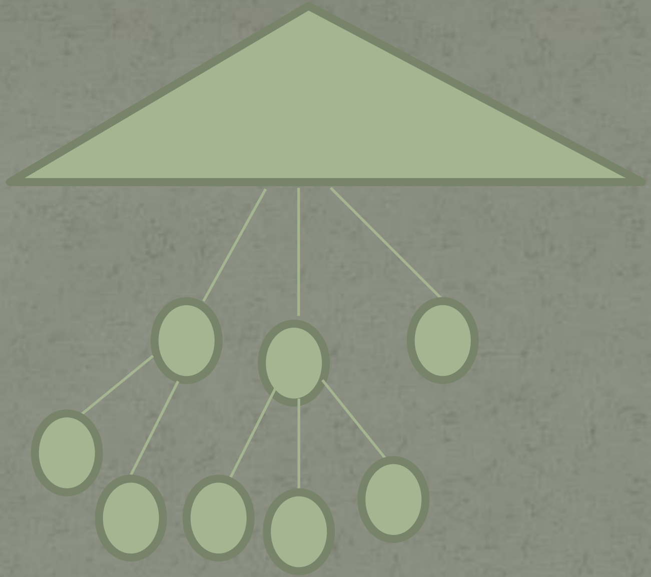
Well founded Ordering + Ordinals(2)

- We denote $>_{or}$ to be the well order on ordinals
- Fact: any ordinal $\alpha < \epsilon_0, \alpha \neq 0$ can be uniquely represented in Cantor Normal Form as a sum $\omega^{\alpha_1} + \cdots + \omega^{\alpha_n}$ where $\alpha_1 \geq_{or} \ldots \geq_{or} \alpha_n$
- Let $\alpha = \omega^{\alpha_1} + \cdots + \omega^{\alpha_n}$ and $\beta = \omega^{\alpha_{n+1}} + \cdots + \omega^{\alpha_{n+m}}$. The natural sum is defined as $\alpha \oplus \beta = \omega^{\alpha_{\pi(1)}} + \cdots + \omega^{\alpha_{\pi(n+m)}}$ when π is a permutation such that $\alpha_{\pi(1)} \geq_{or} \ldots \geq_{or} \alpha_{\pi(n+m)}$
- We write αn as an abbreviation of α+…+α (n times)

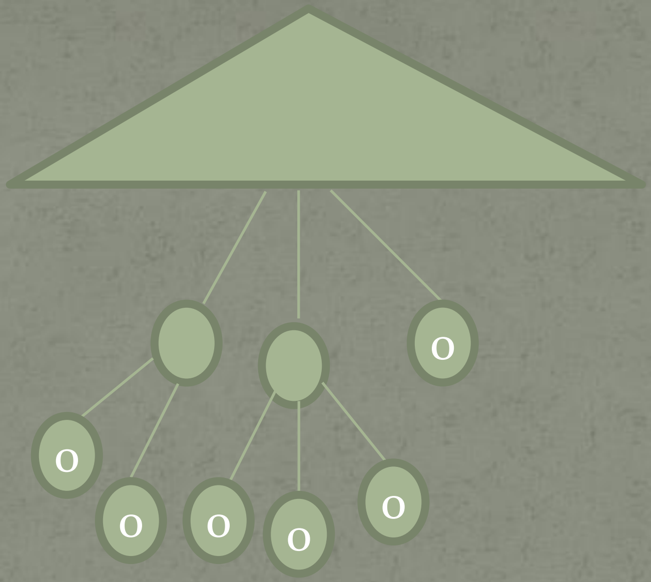
Hercules Prevails(1)

- Hercules win is only a question of time and not a question of if.
- We assign an ordinal $\alpha < \epsilon_0$ to each hydra by the following recursive algorithm:
- Assign 0 to each leaf
- Each internal node is assigned the natural sum of it's children, i.e $\omega^{\alpha_1} \oplus \cdots \oplus \omega^{\alpha_n}$ when α_i are the ordinals already assigned to the children.
- Hydra is represented by the ordinal assigned to it's root.

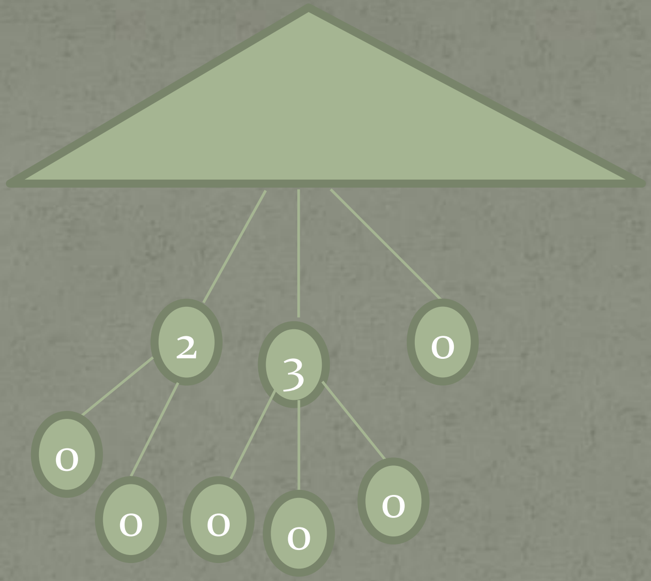
Example(1)



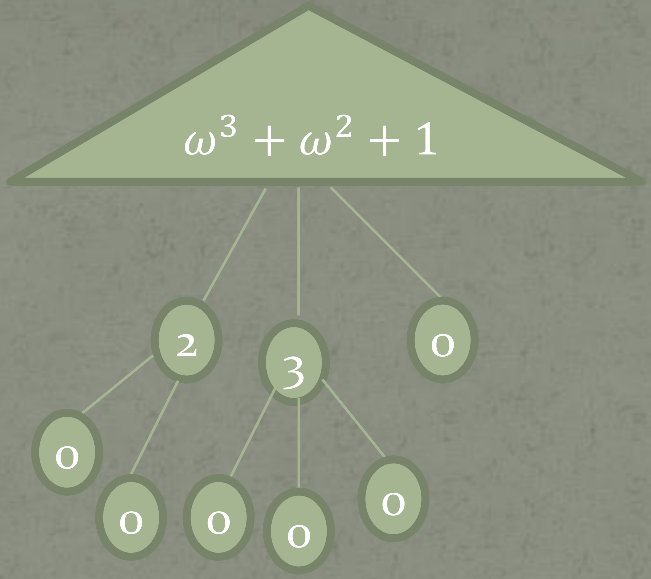
Example(2)



Example(3)



Example(4)



Hercules Prevails(2)

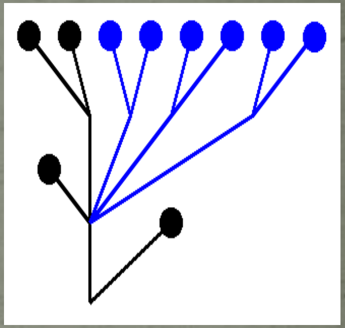
- Let (H, n) denote a configuration of the game
- Let S be some strategy taken by Hercules, then we denote $(H)_n^S$ to be the resulting Hydra if strategy S is used against H at stage n. This means that the next configuration when S is used is $((H)_n^S, n+1)$

Lemma 1 (Kirby & Paris): For any strategy S, Hydra H and natural number n, we obtain $H >_{or} (H)_n^S$
(that is, the ordinal representing the hydra in step n is bigger than the ordinal representing the hydra at step n+1 after using strategy S)

Hercules Prevails(3)

Proof:

- If Hercules chops a head with no grandparent node, this means it is attached to the root, which means the total is decreased by 1.
- If the head cut does have a grandparent node then if we are in stage n, then we replace the term of the ordinal (of the grandparent node) $\omega^{\alpha+1}$ with $\omega^{\alpha}n$ which is a smaller ordinal.



Conclusion: at each step, the ordinal decreases.

Hercules Prevails(4)

Theorem 1: Every strategy is a winning strategy.

Proof: let's assume otherwise. Then there must be some strategy S which makes Hercules lose.

This means that his fight with Hydra never terminates, i.e we have an infinite number of configurations

$C_1, C_2, \dots$. By the Lemma introduced in the previous slide, $C_1 >_{or} C_2 >_{or} C_3 >_{or} \dots$, which means we have found an infinite descending sequence of ordinals, in contradiction to $>_{or}$ being well founded

Defining a Strategy

- The following strategy has been defined "standard".

For an ordinal (Hydra) $\alpha < \epsilon_0$ and some $n \in N$ (the stage) we associate an ordinal $\alpha_n < \epsilon_0$(the next Hydra)

$$\alpha_n = \begin{cases} 0 & \text{if } \alpha = 0 \\ \beta & \text{if } \alpha = \beta + 1 \\ \beta + \omega^{\gamma} n & \text{if } \alpha = \beta + \omega^{\gamma+1} \\ \beta + \omega^{\gamma_n} & \text{if } \alpha = \beta + \omega^{\gamma} \text{ and } \gamma \text{ is a limit ordinal} \end{cases}$$

The Standard Hydra Battle

- A hydra is an ordinal $\alpha < \epsilon_0$. The Hydra battle is a sequence of configurations .
- A configuration is a pair (α, n) when α denotes a hydra and $n \geq 1$ is the current step.
- The next configuration in the standard strategy is $(\alpha_n, n + 1)$

Beyond Peano(1)

We now turn to prove that the natural numbers cannot prove the termination of the standard Hydra Battle.

- The Hardy functions $(H_\alpha)_{\alpha<\epsilon_0}$ are defined as follows:

$$H_0(n) = n$$
$$H_\alpha(n) = H_{\alpha_n}(n+1)\ when\ \alpha > 0$$

Beyond Peano(2)

- Hydra functions:

The related Hydra functions $(L_\alpha)_{\alpha<\epsilon_0}$ are counting the length of the standard Hydra battle starting at stage n with hydra represented by ordinal α.

$$L_0(n) = 0$$
$$L_\alpha(n) = L_{\alpha_n}(n+1) + 1 \; when \; \alpha > 0$$

Beyond Peano(3)

Lemma 2: For any $\alpha < \epsilon_0$ and for any $n \in N$

$H_\alpha(n) = H_{L_\alpha(n)}(n) = n + L_\alpha(n)$

Proof: we first prove that $H_{L_\alpha(n)}(n) = n + L_\alpha(n)$

Claim: for a finite m, $H_m(n) = m + n$

Proof: by induction on m:

Base: for m=0 we have $H_0(n) =_{definition} n$

Step: for m+1 we have

$H_{m+1}(n) = H_m(n+1) =_{i.h} m + n + 1.$

Since Hercules defeats hydra in a finite number of steps, $L_\alpha(n)$ is finite, and $H_{L_\alpha(n)}(n) = n + L_\alpha(n)$

Beyond Peano(4)

Lemma 2: For any $\alpha < \epsilon_0$ and for any $n \in N$

$H_\alpha(n) = H_{L_\alpha(n)}(n) = n + L_\alpha(n)$

Proof: we now turn to prove $H_\alpha(n) = H_{L_\alpha(n)}(n)$

by induction on α:

Base:

$\alpha = 0$: $H_{L_0(n)}(n) = H_0(n)$

Step:

Denote $\gamma = \alpha_n$,

$H_\alpha(n) =_{H-def} H_\gamma(n+1) =_{i.h}$

$H_{L_\gamma(n+1)}(n+1) =_{H-def} H_{L_\gamma(n+1)+1}(n) =_{L-def} H_{L_\alpha(n)}(n)$

Beyond Peano(5)

- We denote PA as Peano Arithmetic.
- A function f is provably recursive in PA if there exists a primitive recursive predicate P and a primitive recursive function(defined before) g such that

  $PA \vdash \forall y_1 \dots \forall y_k \exists x P(y_1, \dots, y_k, x)$ and f satisfies

  $f(n_1, \dots, n_k) = g(\mu_x P(n_1, \dots, n_k, x))$, where μ_x denotes

  the least number operator.

Beyond Peano(6)

- The Hardy class is defined as the smallest class of functions s.t:

1. It contains the zero function, the successor function, all $H_\alpha, s.t\ \alpha < \epsilon_0$ and all projection functions.
2. It is closed under primitive recursion and composition.

Theorem 2(Gaisi Takeuti) : The Hardy class is the class of all provably recursive functions in PA (no proof provided in this lecture)

Beyond Peano(7)

Theorem 3: PA cannot prove termination of the standard Hydra Battle.

Proof: Lets assume otherwise. This can be written as

$PA \vdash \forall \alpha, n \exists m, s.t\ L_\alpha(n) = m.$

In this case

$$P(\alpha, n, m) = (L_\alpha(n) = m)$$
$$f(\alpha, n) = L_\alpha(n)$$
$$g(\alpha, n, m) = \mu_m(L_\alpha(n) = m)$$

This means that $\forall \alpha < \epsilon_0, \forall n \in N\ L_\alpha(n)$ is

Provably recursive in PA

Beyond Peano(8)

$L_{(\epsilon_0)_n}(n+1)+1$ is provably recursive, and by definition $L_{(\epsilon_0)_n}(n+1)+1 = L_{\epsilon_0}(n)$ is also provably recursive. Lemma 2 suggests that $L_{\epsilon_0}(n)+n = H_{\epsilon_0}(\text{n})$, which would imply that $H_{\epsilon_0}(n)$ is provably recursive in contradiction to theorem 2. ■