Rank Bound for Depth-3 Identities

Nitin Saxena (Hausdorff Center for Mathematics, Bonn)

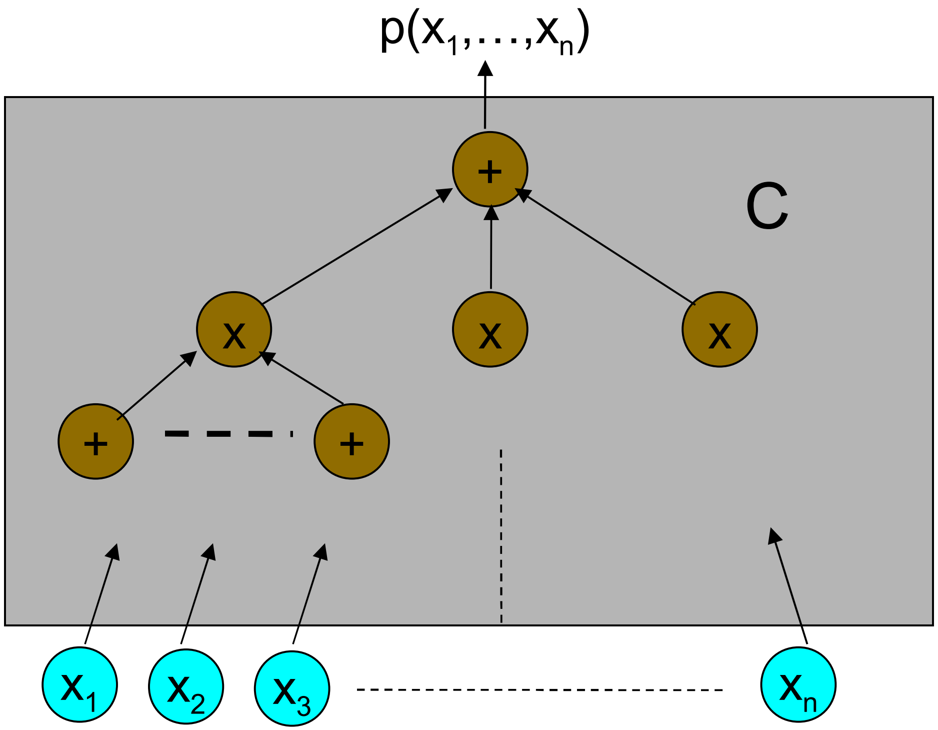
Joint work with

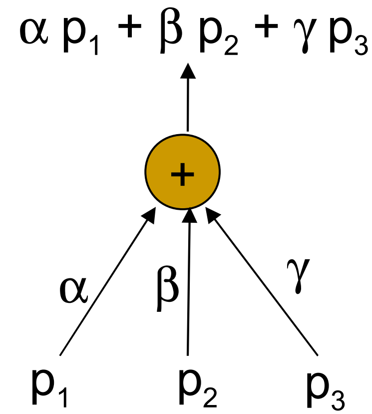
C. Seshadhri (IBM Almaden Research Center, San Jose)

The problem of PIT

- Polynomial identity testing: given a polynomial $p(x_1,x_2,\ldots,x_n)$ over F, is it identically zero?
  - *All* coefficients of $p(x_1,\ldots,x_n)$ are zero.
  - $(x+y)^2 - x^2 - y^2 - 2xy$ is identically zero.
  - So is: $(a^2+b^2+c^2+d^2)(A^2+B^2+C^2+D^2)$
    $- (aA+bB+cC+dD)^2 - (aB-bA+cD-dC)^2$
    $- (aC-bD-cA+dB)^2 - (aD-dA+bC-cB)^2$
  - $x(x-1)$ is NOT identically zero over F_2.

Circuits: Blackbox or not





We want algorithm whose running time is polynomial in size of the circuit.

- Non blackbox: can analyze structure of C
- Blackbox: cannot look *inside* C
  - Feed values and see what you get

A simple, randomized test

If output is 0, we guess it is identity.

Otherwise, we know it isn't.

- [Schwartz '80, Zippel '79] This is a randomized blackbox poly-time algorithm.
- (Big) open problem: Find a deterministic polynomial time algorithm.
  - We would really like a black box algorithm

Why?

- Come on, it's an interesting mathematical problem. Do you need a further reason?
- [Impagliazzo Kabanets '03] Derandomization implies circuit lower bounds for permanent

- [AKS '02] Primality Testing ; $(x + a)^n – x^n - a = 0 \pmod{n}$
- [L '79, MVV '87] Bipartite matching in NC?...
- Many more

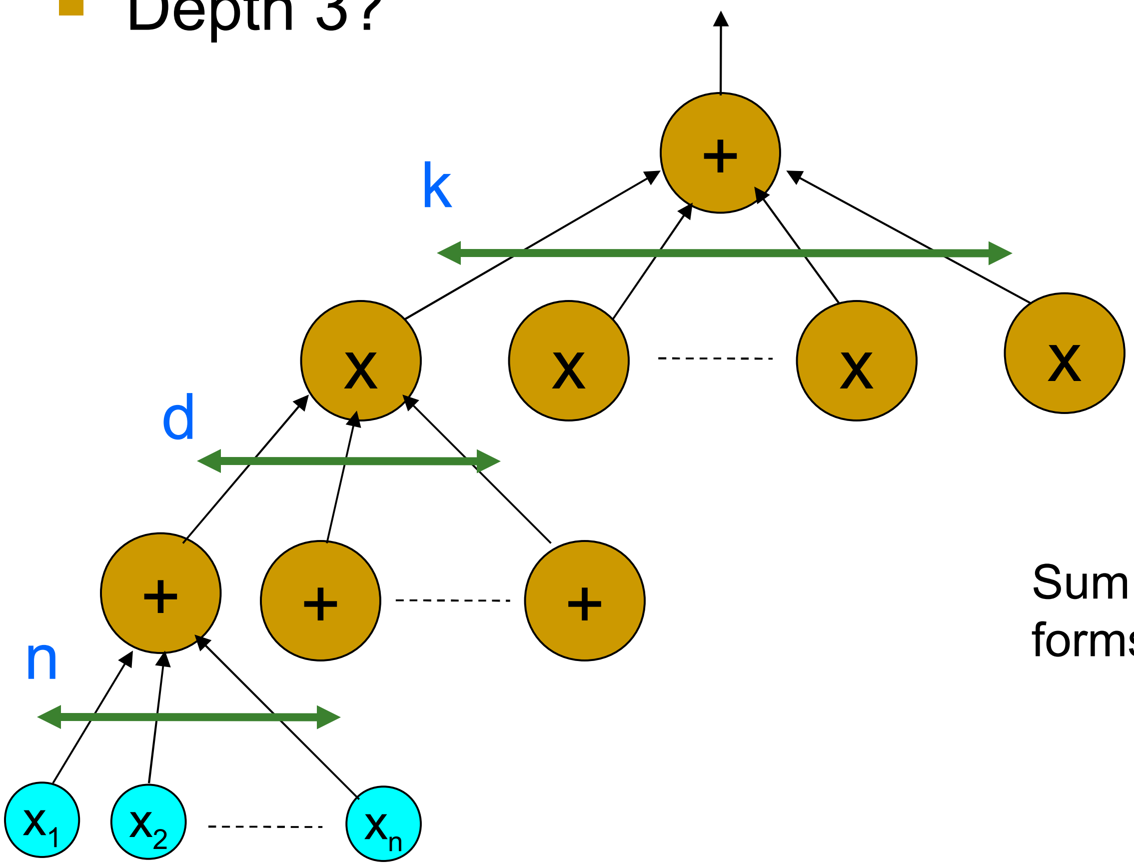
What do we do?

George Pólya 1887-1985

If you can't solve a problem, then there is an easier problem you *can* solve. Find it.

Get shallow results

- Let's restrict the depth and see what we get
- Depth 2? Non-blackbox trivial!
  - [GKS '90, BOT '88] Polytime & blackbox
- Depth 3?

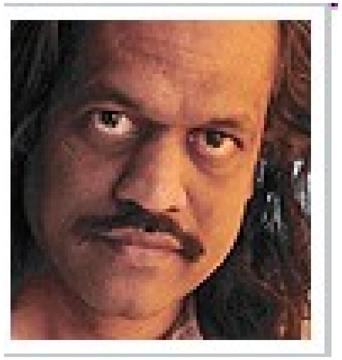


$$C = \sum_{i=1}^{k} \prod_{j=1}^{d} L_{ij} = \sum_{i=1}^{k} T_i$$

Sum of products of kd linear forms in n variables

Shallowness is not so bad!

M. Agrawal



V. Vinay

- They say…
- [AV '08] Chasm at Depth 4!
- If you can solve blackbox PIT for depth 4, then you've "solved" it for all depths.

Shallowness is not so bad!

The two main ideas involved are......

- [AJMV '98] Any circuit C computing a polynomial $p(x_1,...,x_n)$ of degree d can be converted into a depth O(log d) circuit C'.
- [AV '08] Few top layers of C' are collapsed to get a depth-2 circuit. The same is done to the remaining bottom layers of C'.
- This yields a depth-4 circuit C'' with only a subexponential blowup.

The past… [redacted] … of Depth-3

- [Dvir Shpilka '05] Non-blackbox poly(n)exp((log d)^k) time algorithm
- [Kayal Saxena '06] Non-blackbox poly(n,d^k) time algorithm

[redacted]

- [Karnin Shpilka '08] Blackbox poly(n)exp((log d)^k)
- [Us] Blackbox poly(n)exp(k^3 log^2 d)

The rank

$$M = \begin{array}{cc} & n \\ kd & \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots\cdots & \alpha_n \\ & & & \\ & & & \\ & & & \end{bmatrix} \end{array}$$

Rank(C) = Rank (M)

$$C \equiv \sum_{i=1}^{k} \prod_{j=1}^{d} L_{ij}$$

$$L_{ij} = \sum_{r=1}^{n} \alpha_r x_r$$

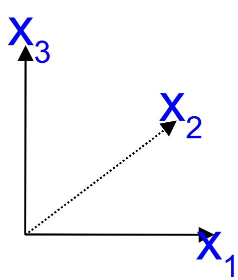
n-dim vector over F

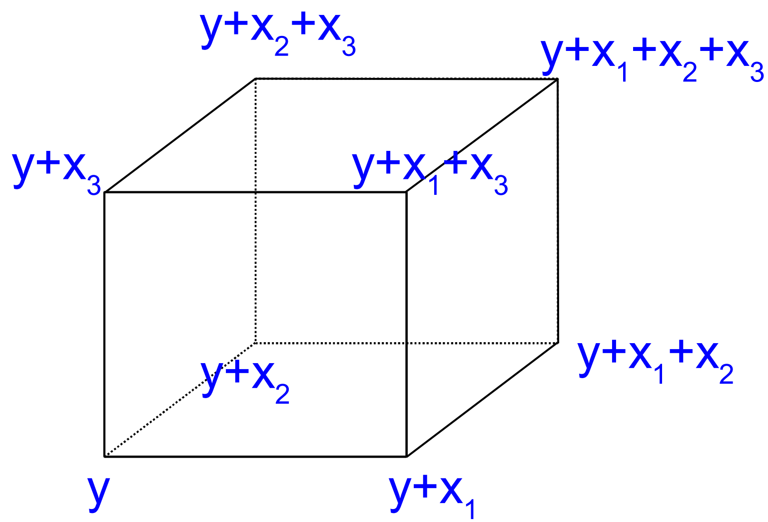
- Introduced by [DS '05]: fundamental property of depth 3 circuits
- [DS '05] Rank of *simple minimal* identity < $(\log d)^{k-2}$ (compare with kd)
- How many independent variables can an identity have?
  - An identity is very constrained, so few degrees of freedom

Exemplary Example

Here is *the highest rank* depth-3, fanin-3 example over Reals.

- $y(y+x_1+x_2)(y+x_2+x_3)(y+x_3+x_1) - (y+x_1)(y+x_2)(y+x_3)(y+x_1+x_2+x_3)$
  $+ x_1x_2x_3(2y+x_1+x_2+x_3) = 0$
- It is of rank 4.
- It is easy to see the geometry behind this identity:





What we did

- Rank of depth 3 simple, minimal identity < $k^3 \log d$
  - There is identity with rank ($k \log d$), so this is almost optimal
  - Let P be a nonzero poly generated by depth 3 circuit. Then rank of linear factors of P is at most $k^3 \log d$
- So [KS '08] implies det. blackbox $\exp(k^3 \log^2 d)$ test
- We develop techniques to study depth 3 circuits over any field.
  - Probably more interesting/important than result

- [Kayal Saraf '09] If base field is *reals*, rank < k^k

Be simple and minimal

- Depth-3: $C = T_1 + T_2 + \ldots + T_k$
- Simplicity: no common (linear) factor for all T_r's
  - $x_1x_2\ldots x_n - x_1x_2\ldots x_n$ (Rank = n)
- Minimality: no subset of T_r's are identity
  - $x_1x_2\ldots x_nz_1 - x_1x_2\ldots x_nz_1 + y_1y_2\ldots y_nz_2 - y_1y_2\ldots y_nz_2$
    (Rank = 2n+2)
- We give poly-time algo that returns small basis or gives obstruction

Top fanin k=3

- $C = T_1 + T_2 + T_3 = \Pi L_i + \Pi M_j + \Pi N_k = 0$
- [AB '99, AKS '02, KS '06] Go modulo!

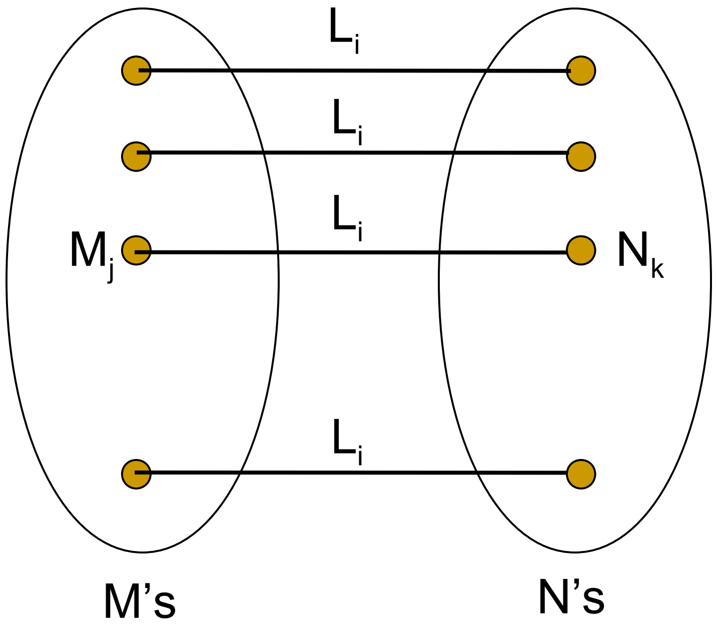
$$\prod L_i + \prod M_j + \prod N_k = 0$$

Vanishes! $\longrightarrow$ $$\boxed{\prod L_i} + \prod M_j + \prod N_k = 0 \pmod{L_1}$$

$$\prod M_j = -\prod N_k \pmod{L_1}$$

- By unique factorization, there is 1-1 mapping between M's and N's (they are same upto constants)
- This is the L_1 matching.

The L_i matchings



$$M_j \equiv \alpha N_k \pmod{L_i}$$

$$M_j = \alpha N_k + \beta L_i$$

Suppose $\alpha = 0$

$$M_j = \beta L_i$$

$$\prod L_i + \prod M_j + \prod N_k = 0$$

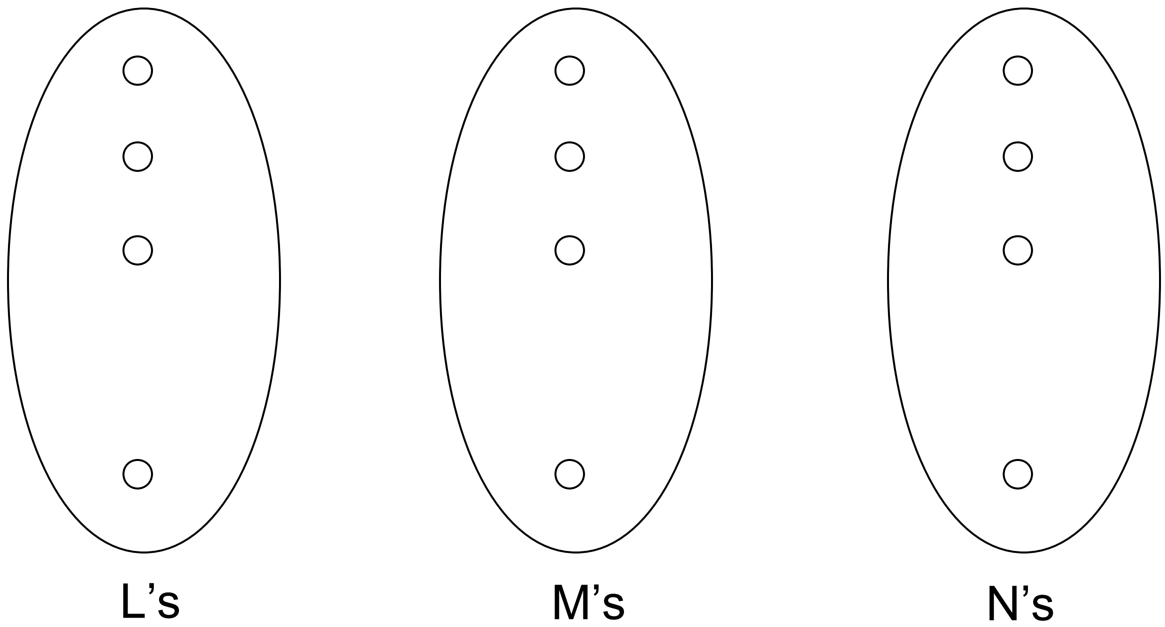
So L_i is common factor.

Circuit is not simple!

So $\alpha,\beta \neq 0$

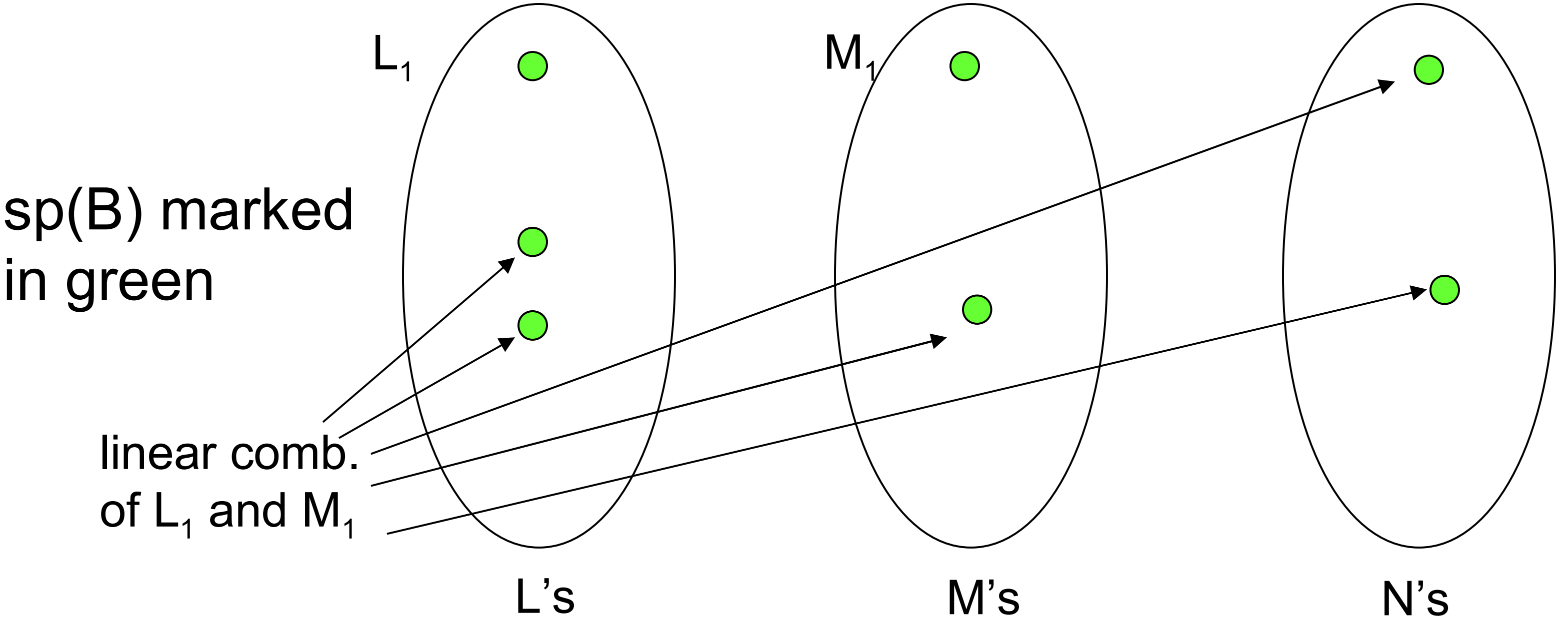
- For every L_i, the M's and N's have a perfect matching
  - Always non-trivial linear combinations

The spanning procedure



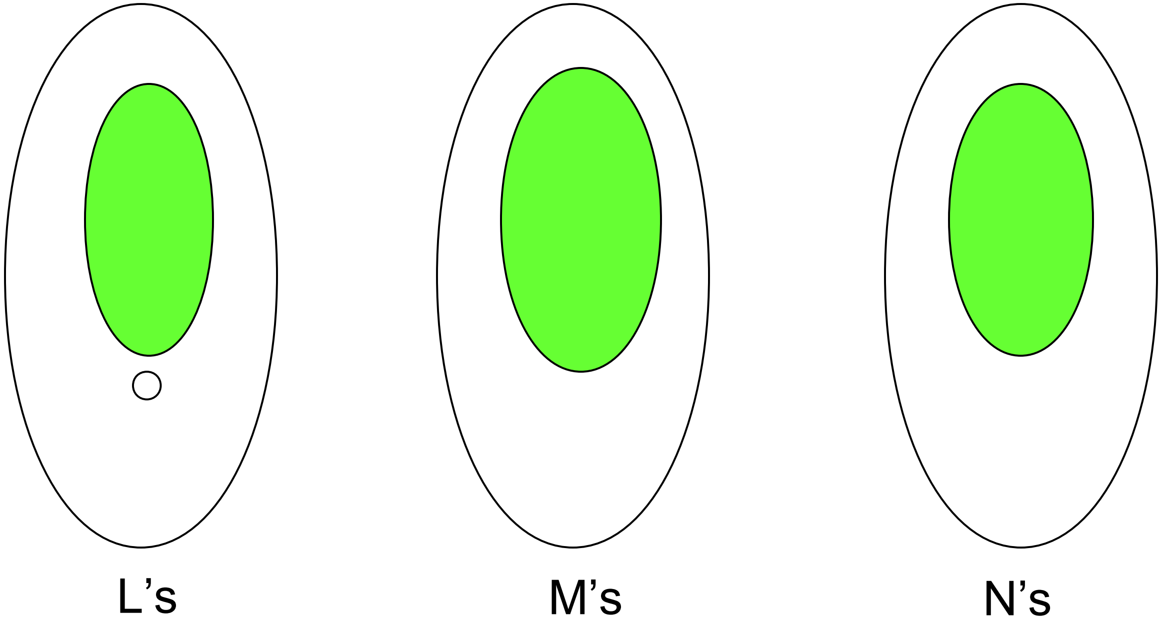
- We iteratively build a basis B.
  - sp(B) is set of forms spanned by B
- Start with B = $\{L_1, M_1\}$

The spanning procedure



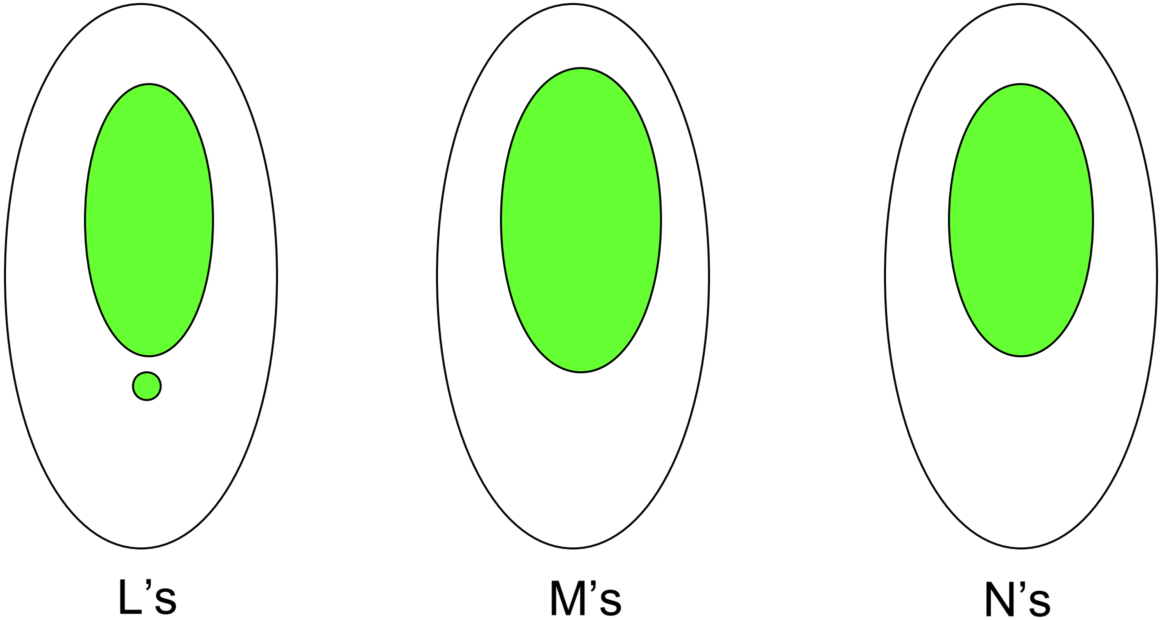
- Start with $B = \{L_1, M_1\}$

The spanning procedure



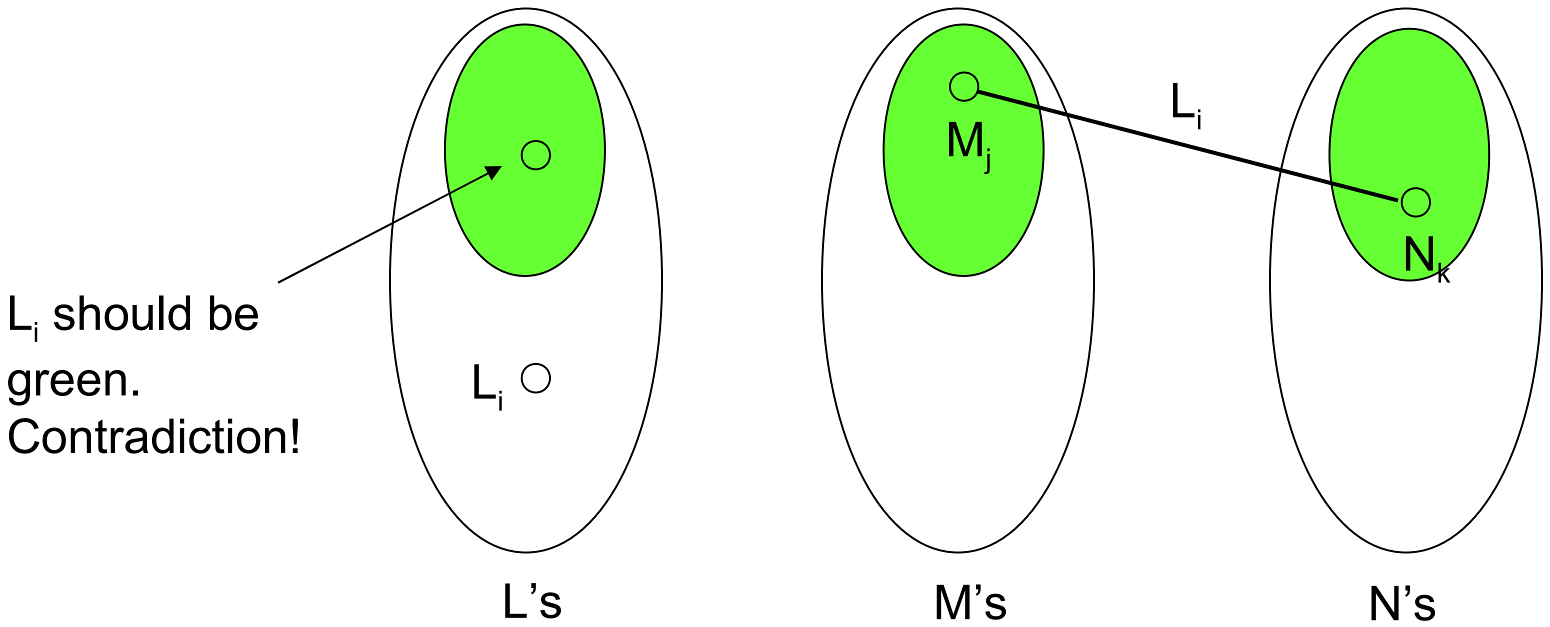
- Start with B = $\{L_1, M_1\}$
- Choose L_2 outside sp(B). Add it to B.

The spanning procedure



- Start with B = {L_1,M_1}
- Choose L_2 outside sp(B). Add it to B.
  - Update sp(B) and repeat until all forms are spanned
- Rank bound = #rounds + 1

The $\log_2 d$ bound

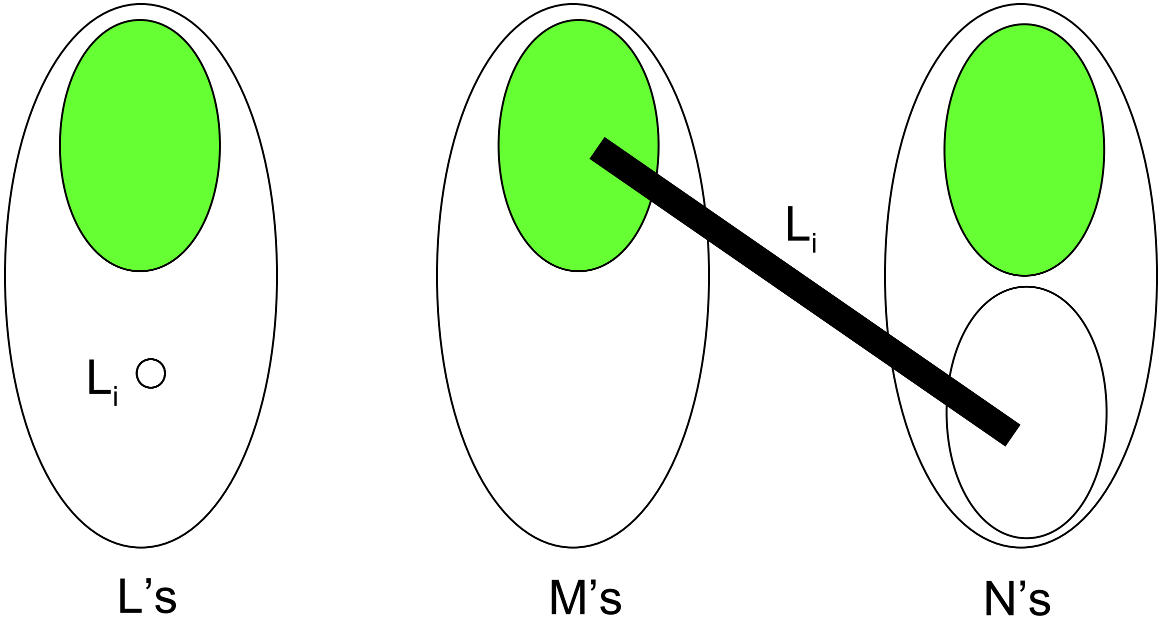


- Claim: After every round, # of green M's doubles

$$M_j = \alpha N_k + \beta L_i \quad \Longrightarrow \quad L_i = \beta^{-1} M_j - \beta^{-1} \alpha N_k$$

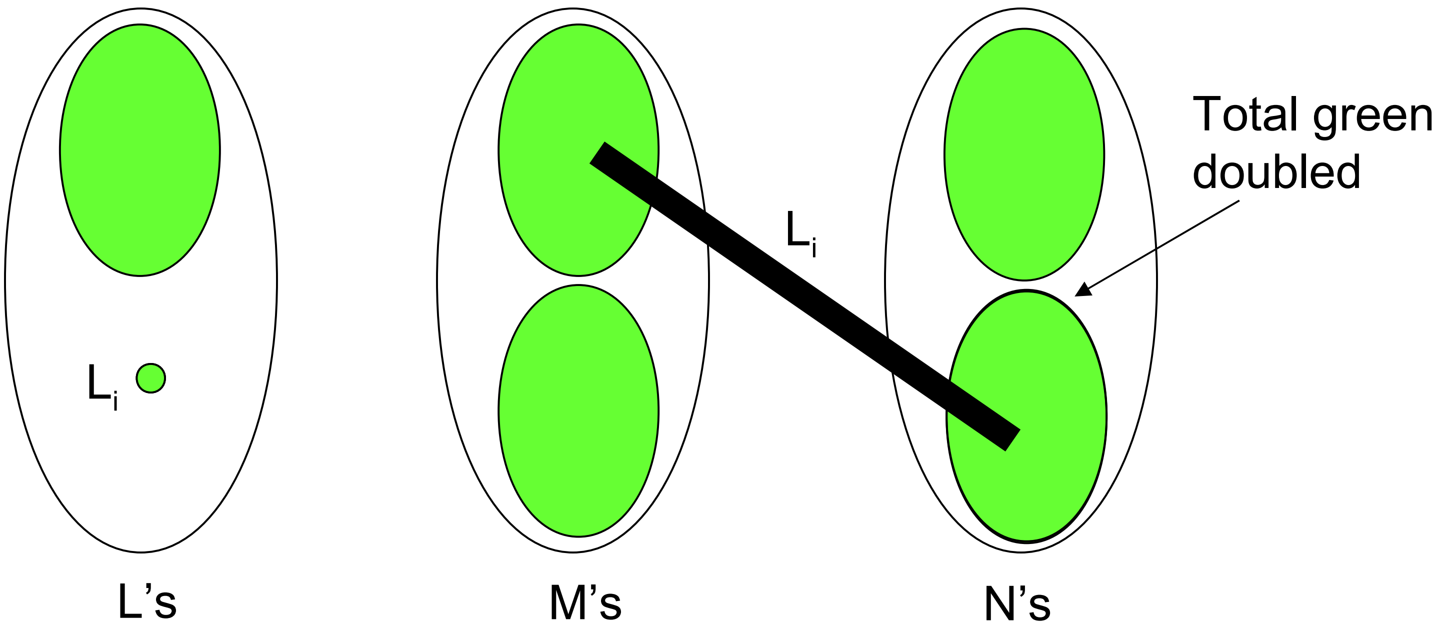
- All L_i neighbors of green part are not green

The $\log_2 d$ bound



- Claim: After every round, # of green M's doubles

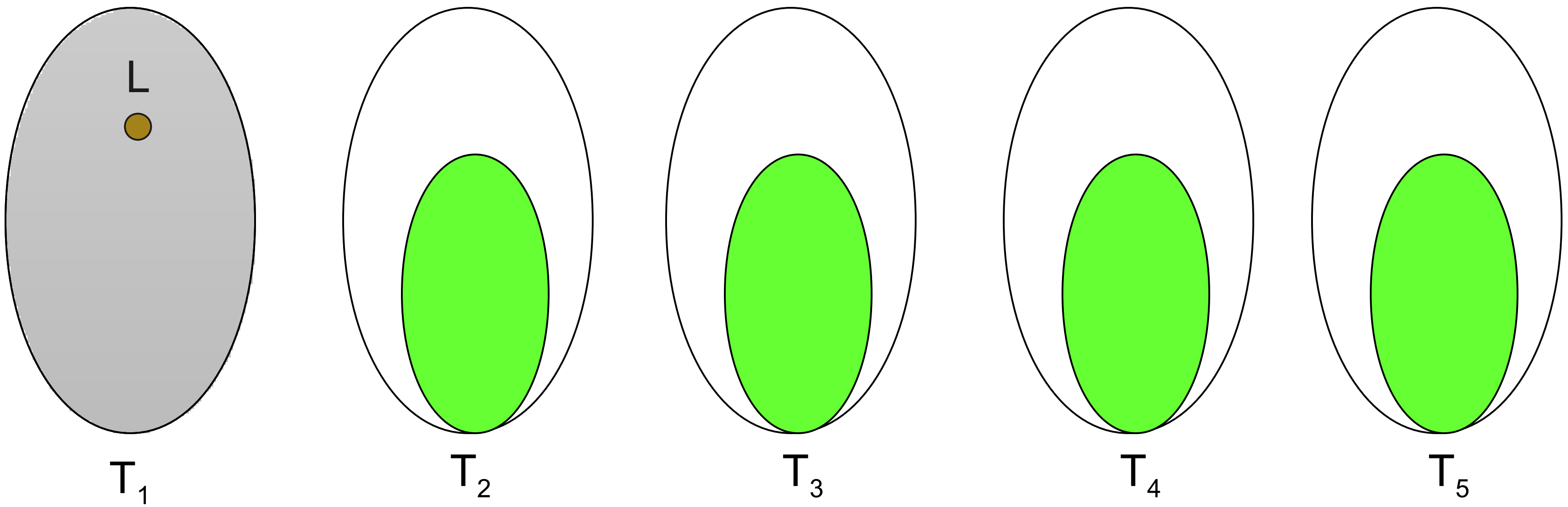
The $\log_2 d$ bound



- Claim: After every round, # of green M's doubles

- Rank bound is ($\log_2 d + 1$)
- Lower bound example has exactly same matching structure (exists for any finite char field)

Larger k: can't induct easily



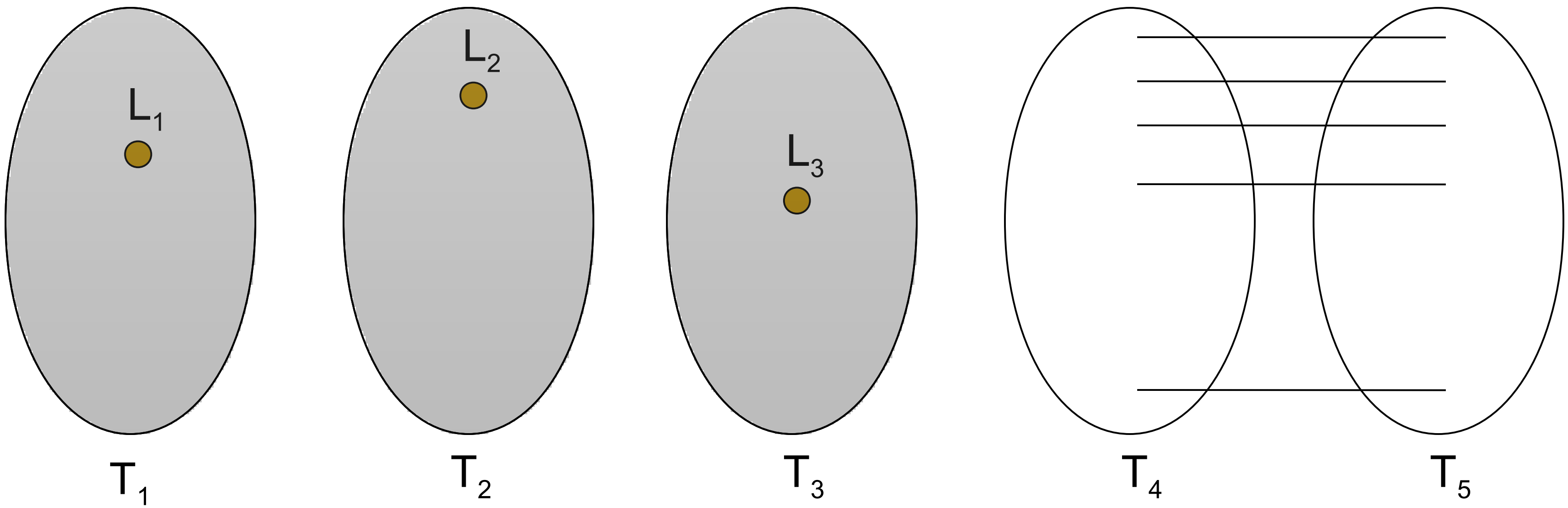
- $C = T_1 + T_2 + T_3 + T_4 + T_5$
- $L \in T_1$. So how about C (mod L)? Top fanin is now 4.
- But C(mod L) may not be simple or minimal any more!
  - $x_1x_2 + (x_3-x_1)x_2 + (x_4-x_2)x_3 - x_3x_4$
  - Going (mod x_1), we get $x_2x_3 + (x_4-x_2)x_3 - x_3x_4$

The ideal way to Matchings

- We saw the power of matchings for k=3
- We extend matchings to ideal matchings for all k
  - Looking at C modulo an ideal, not just a linear form

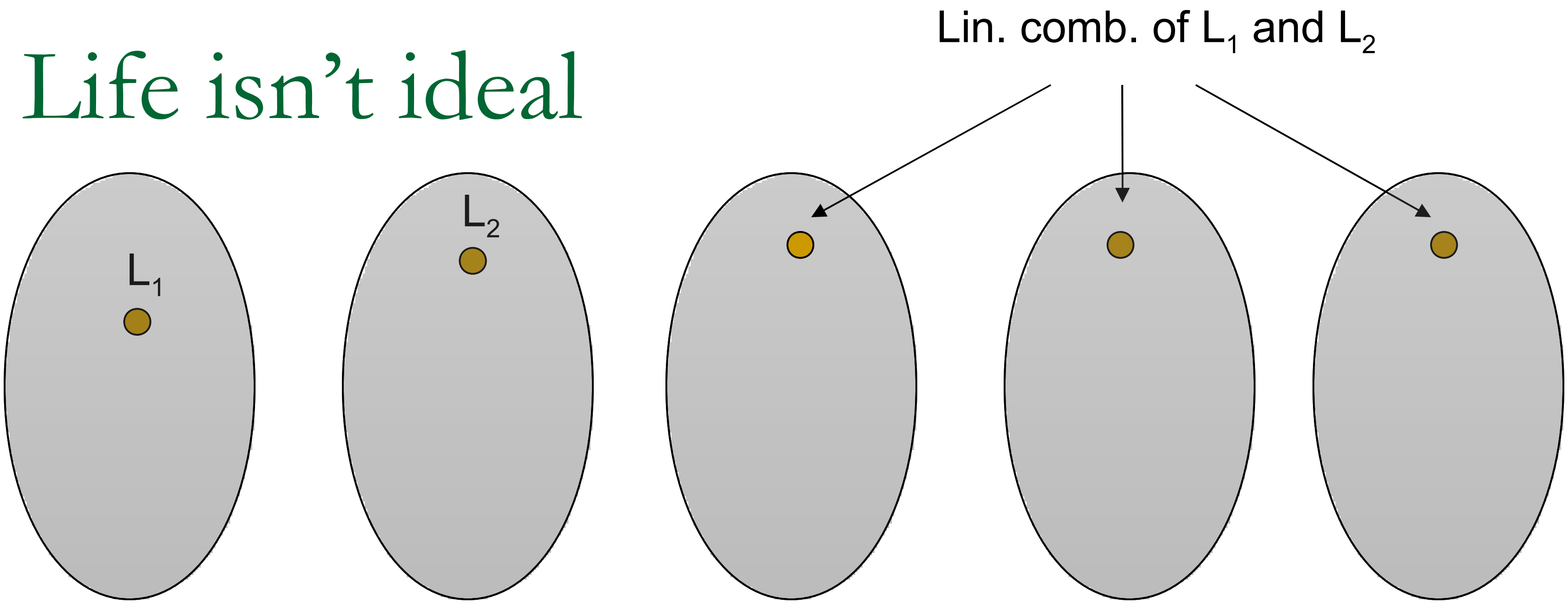
- Use these to construct a spanning procedure as before
  - Find some small set of forms not in sp(B), add them to B, continue
  - The number of rounds of this procedure gives the bound

Ideal matchings



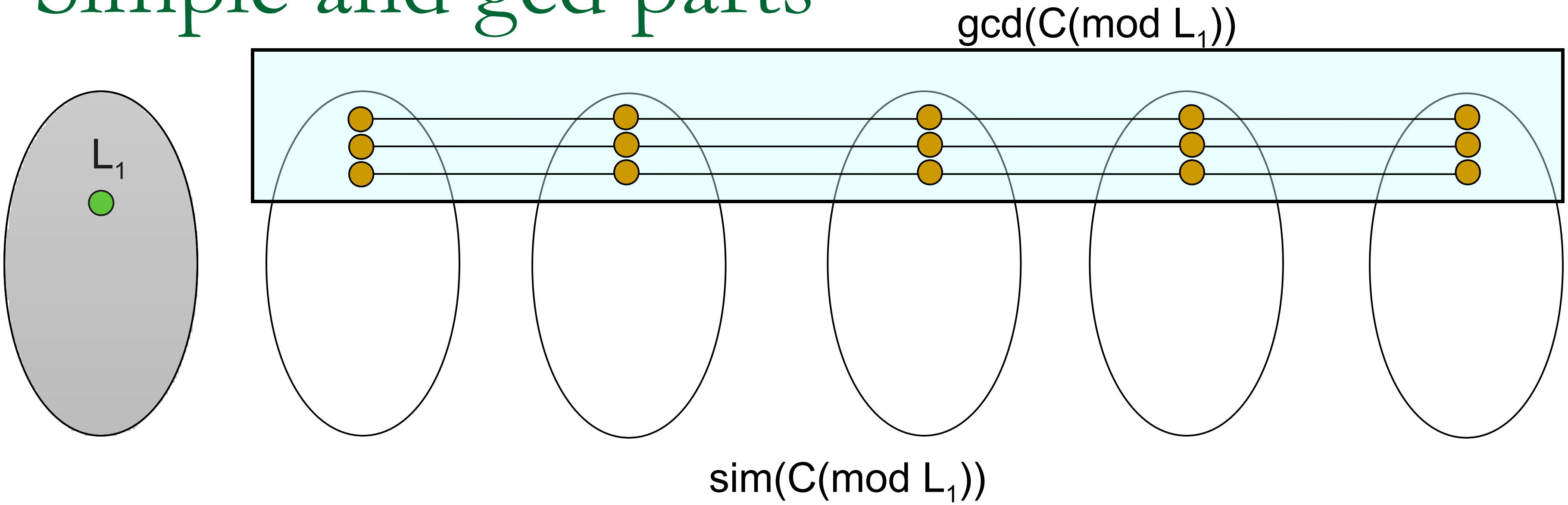
- C (mod L_1, L_2, L_3) or C (mod I)
  - I is ideal $\langle L_1, L_2, L_3 \rangle$
- $T_4 + T_5 = 0$ (mod I)
  - By unique factorization, we get I-matching

Life isn't ideal



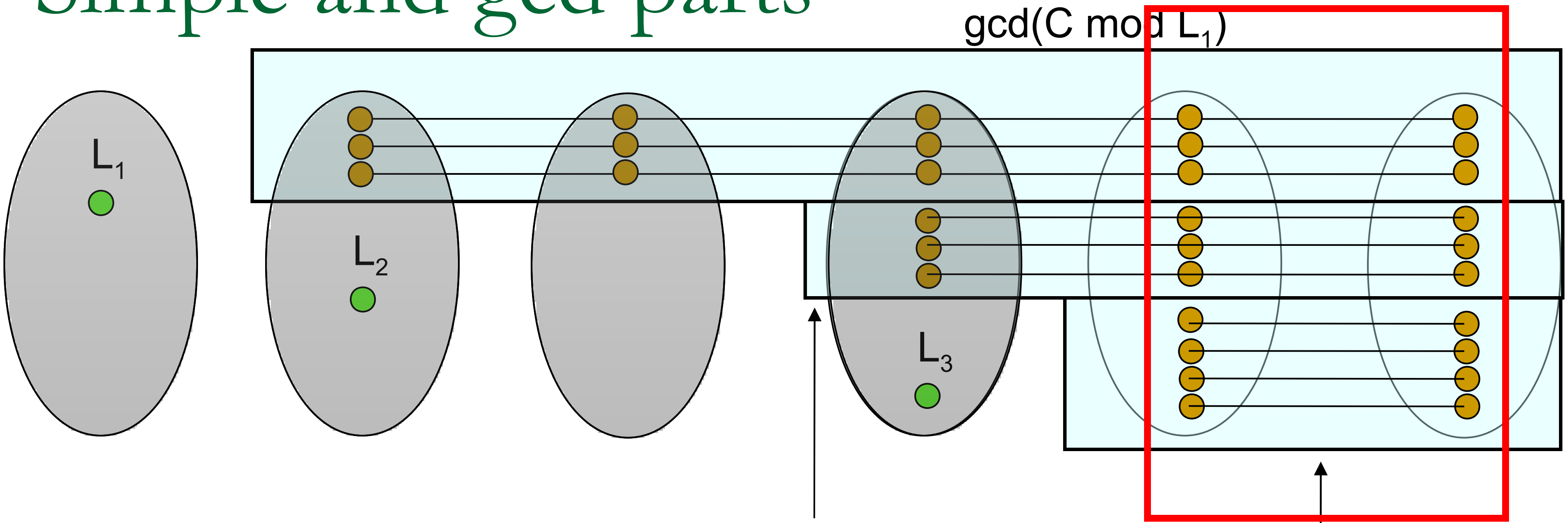
- C (mod L_1, L_2) has no terms
- How can we get a matching?

Simple and gcd parts



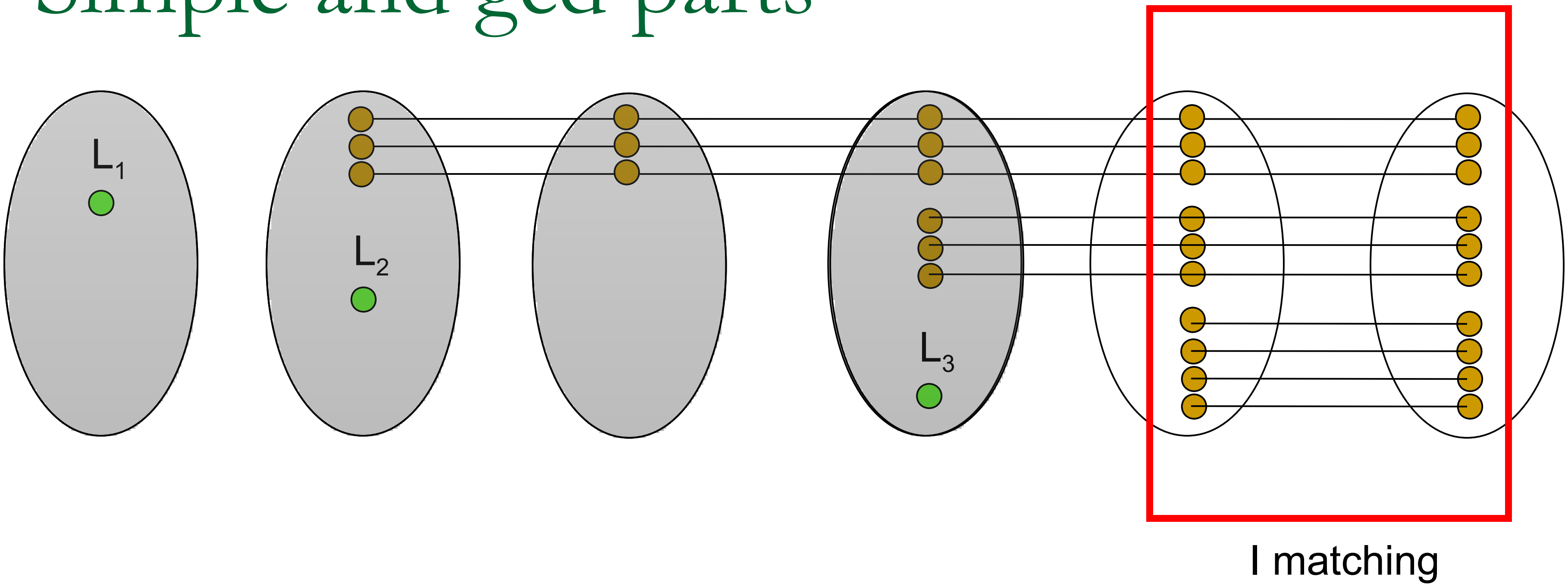
- C (mod l) has gcd part and simple part
- $C = x_1x_2 + (x_3-x_1)x_2 + (x_4-x_2)x_3 - x_3x_4$
- $C \pmod{x_1} = x_2x_3 + (x_4-x_2)x_3 - x_3x_4$
  - So x_3 is gcd(C mod x_1)
  - $x_2 + (x_4-x_2) - x_4$ is sim(C mod x_1)

Simple and gcd parts



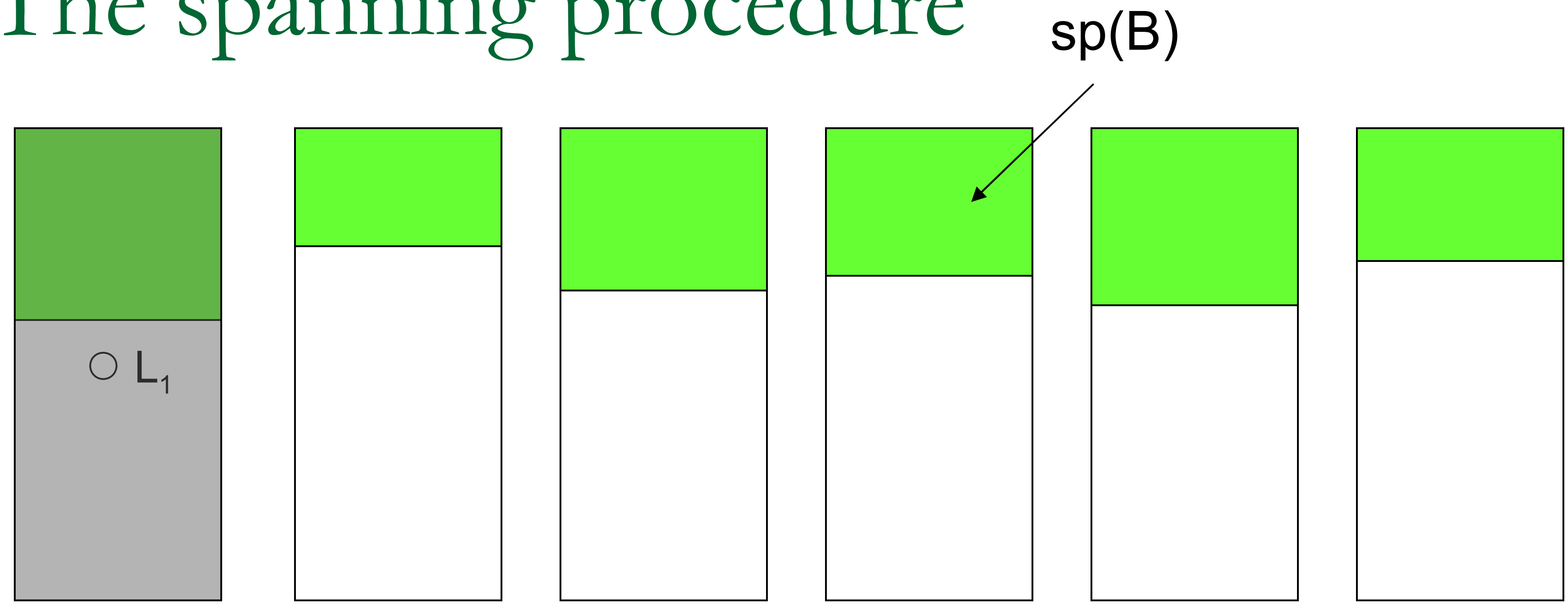
- C (mod I) has gcd part and simple part
- $C = x_1x_2 + (x_3-x_1)x_2 + (x_4-x_2)x_3 - x_3x_4$
- C (mod x_1) = $x_2x_3 + (x_4-x_2)x_3 - x_3x_4$
  - So x_3 is gcd(C mod x_1)
  - $x_2 + (x_4-x_2) - x_4$ is sim(C mod x_1)

Simple and gcd parts

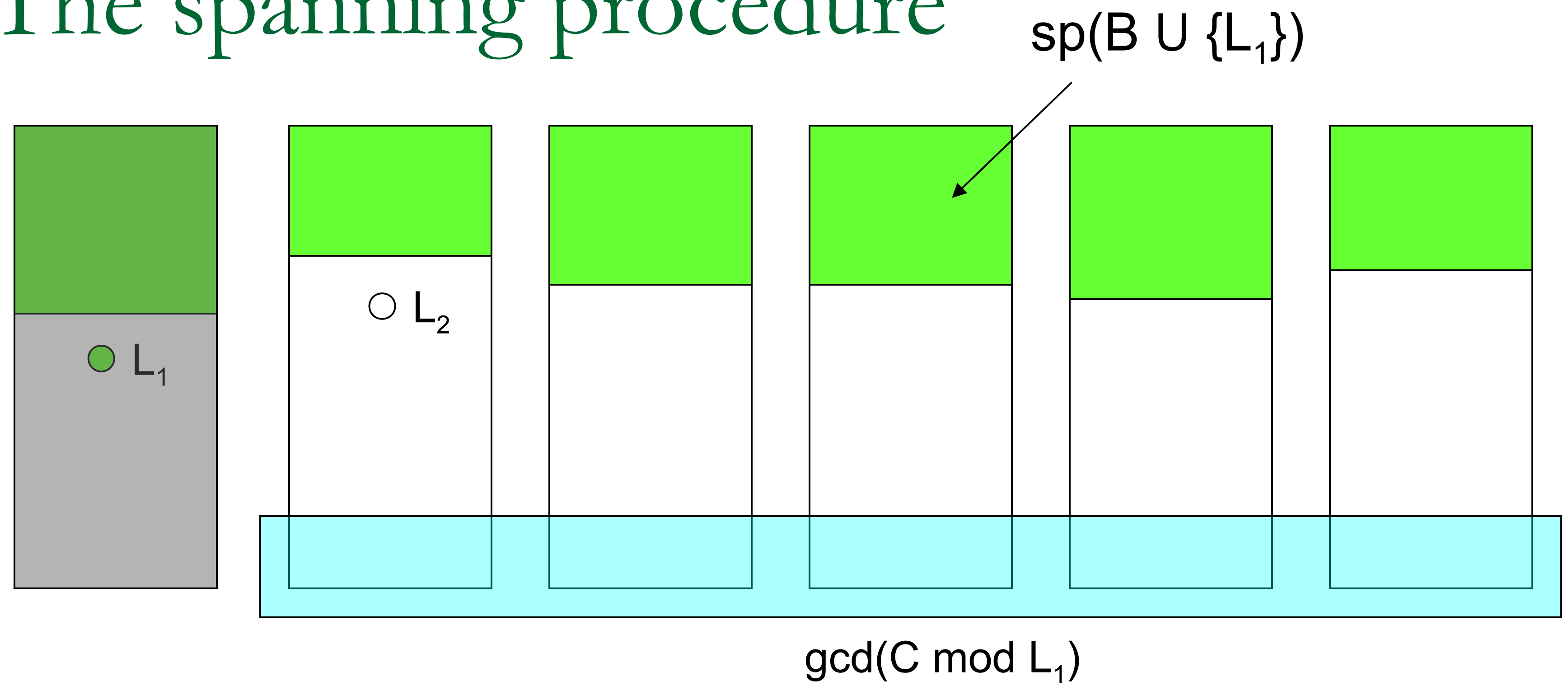


- Let I = <L_1, L_2, L_3>
- Piece together gcd portions
- Eventually, we can’t even use this, but this gives the right idea

The spanning procedure

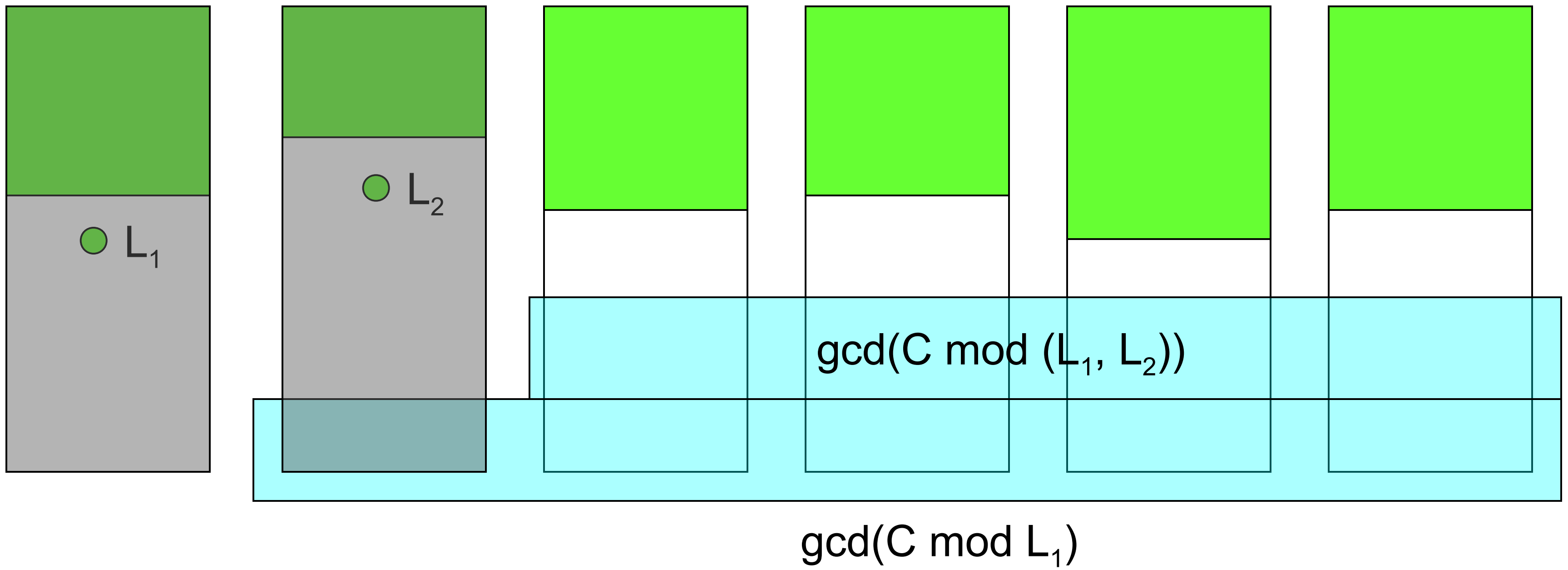


The spanning procedure

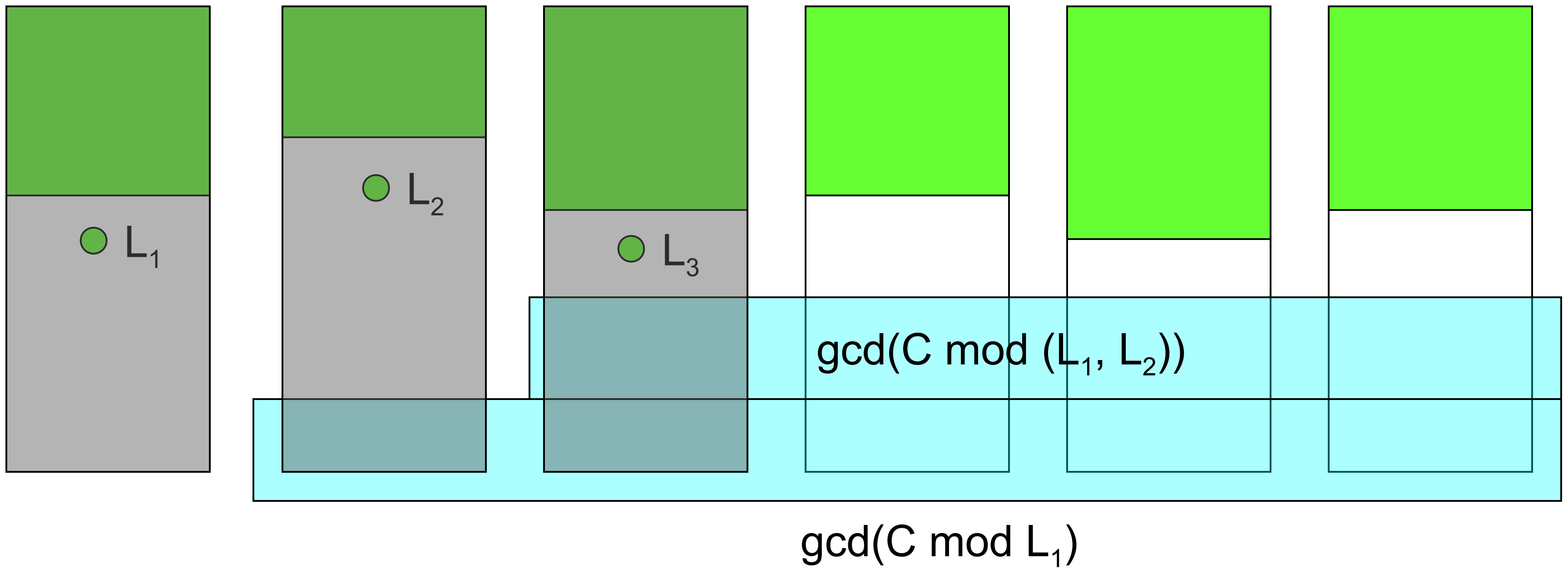


- We want to get ideal I for matching
- Add new form to I, remove gcd (mod I), update sp(B U I), repeat…
- In the end of round, add I to B

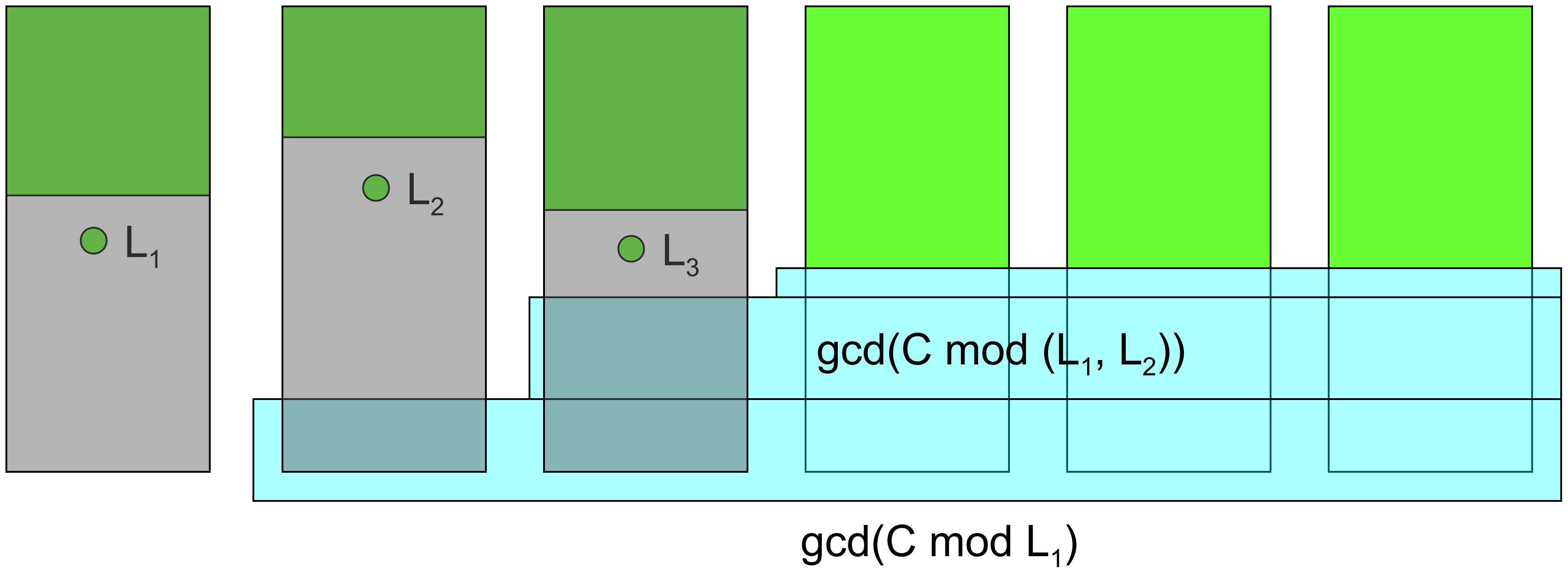
The spanning procedure



The spanning procedure

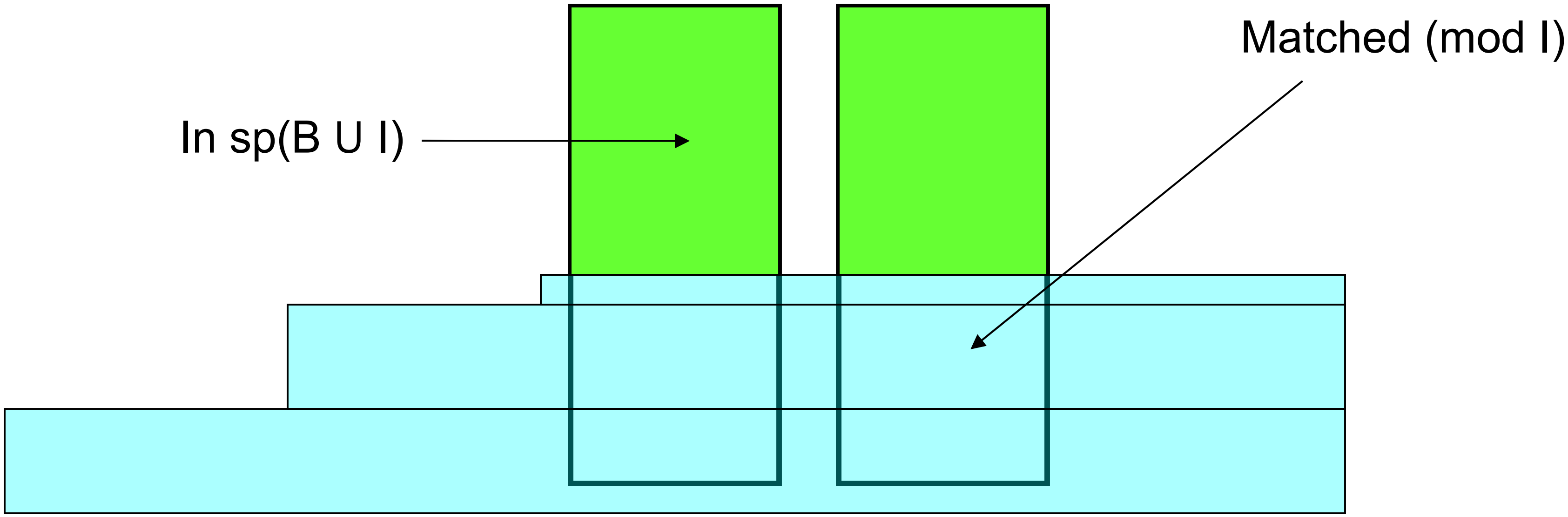


The spanning procedure



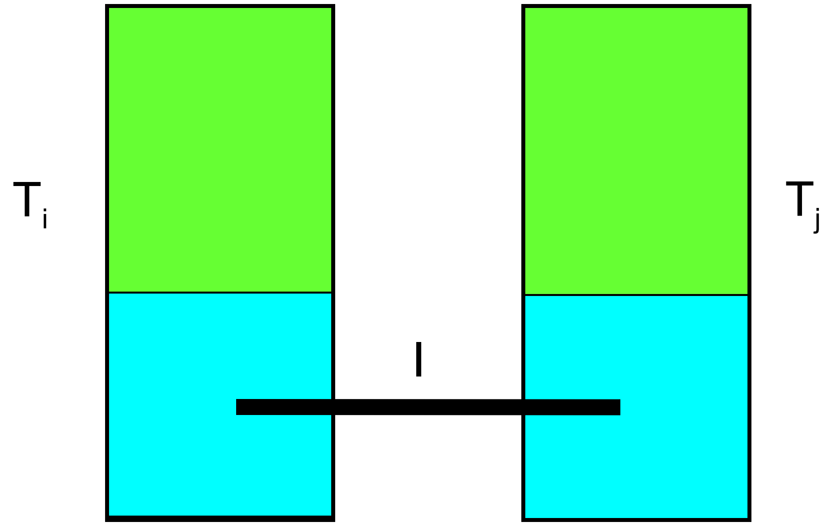
- Progress not possible!
- We only have partial matchings mod I
  - I = <L_1, L_2, L_3>

Partial matchings



- We only get partial matchings at end of round
  - Carefully, we can deal with this
- Rank bound is: k x (# rounds)

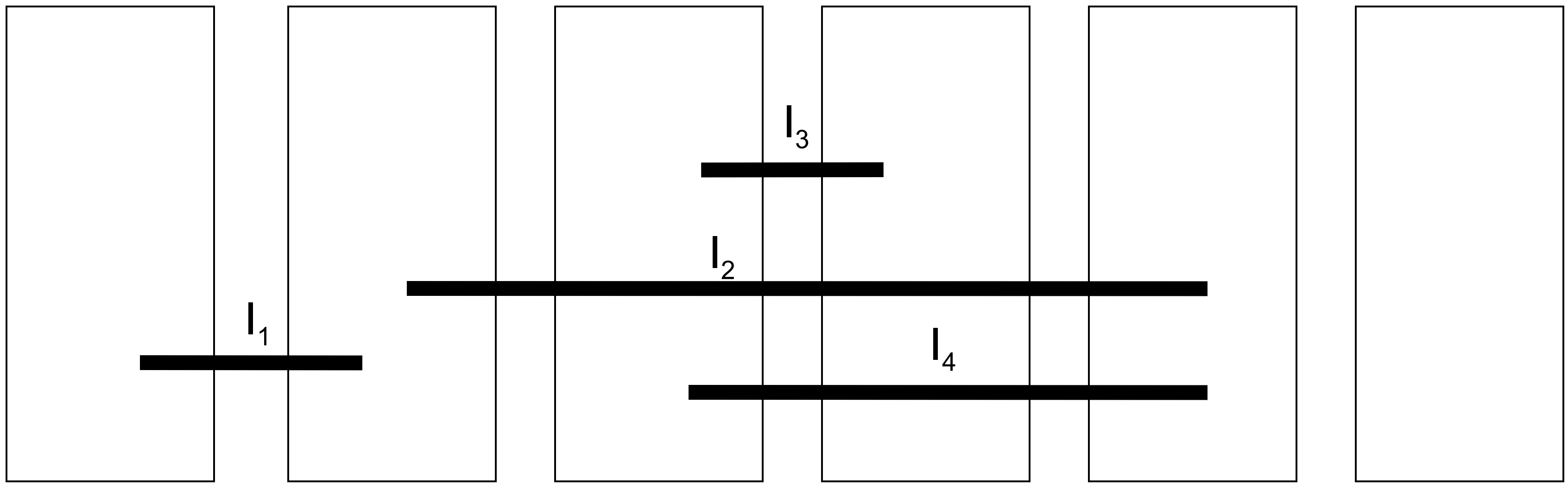
#rounds: Types of matchings



- At beginning of round, we have B
- At end, between two terms we have I-matching

- Type 1: Blue parts have different forms
- Type 2: Blue parts have same forms

Counting Type 1 matchings



- In every round, at least two terms are matched
- If there are more than (k^2 log d) type-1 matchings
  - Pigeonhole argument says one pair (T_i, T_j) is matched more than (log d) times
  - Doubling argument (like k=3) implies that this cannot happen

Counting Type 2 matchings

- This deals with pathological case of same forms getting matched
  - Previous doubling-argument will not work

- That uses a different argument
  - There are at most k of these

- Minimality enters the picture.
  - Algorithmically, we can detect non-minimality

The rank bound

- Thus, #rounds < (k^2 log d) + k
- Rank bound of: k x (k^2 log d + k) = $O(k^3 \log d)$.

In conclusion…

- Interesting matching structures in depth 3 identities
  - Combinatorial view of algebraic properties

- Can we get poly(k) rank when F = R?
  - [Kayal Saraf 2009] get k^k

- What about identity testing for depth 3 circuits? Nothing is known when k is large
  - [DS 2005, KS 2006] use some recursive arguments that get k in exponent
  - Maybe our techniques can get around this…?

A Saxena-Seshadhri paper