Weak Composition and *Polynomial Lower Bounds* for *Kernelization*

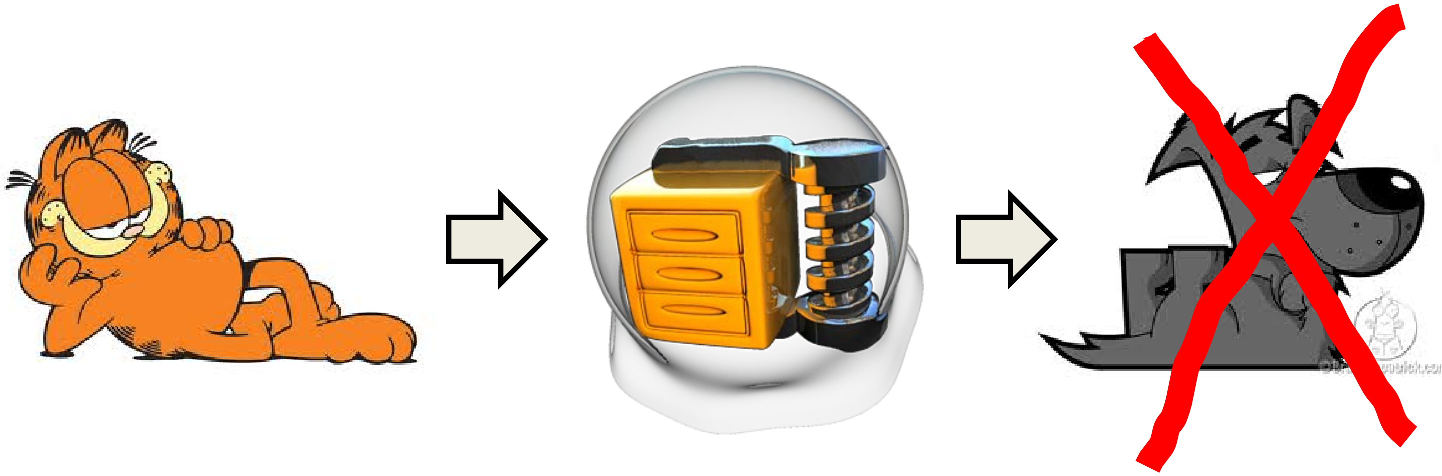
Xi Wu
University of Wisconsin-Madison, US

Joint work with *Danny Hermelin*
Max Planck Institute for Informatics, Saarbrücken, Germany

- From *Merriam-Webster*:

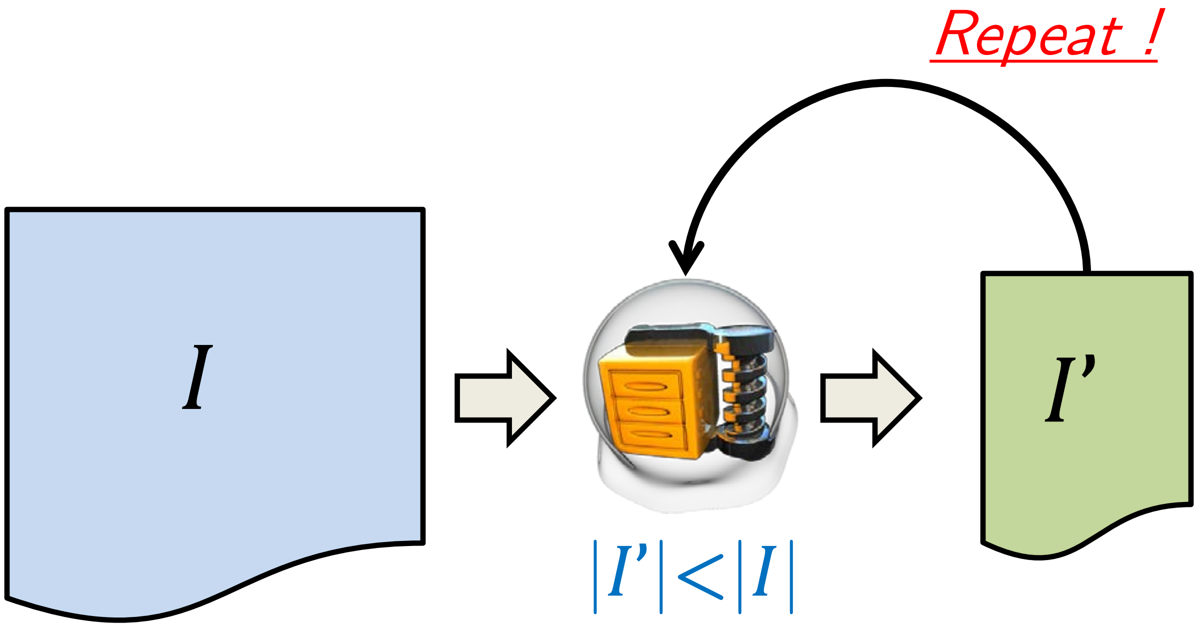
> *compression*: *noun* |*kəm-ˈpre-shən*|
>
> *conversion of data in order to reduce the space occupied or bandwidth required.*

- One point should be added: *keep some properties*.



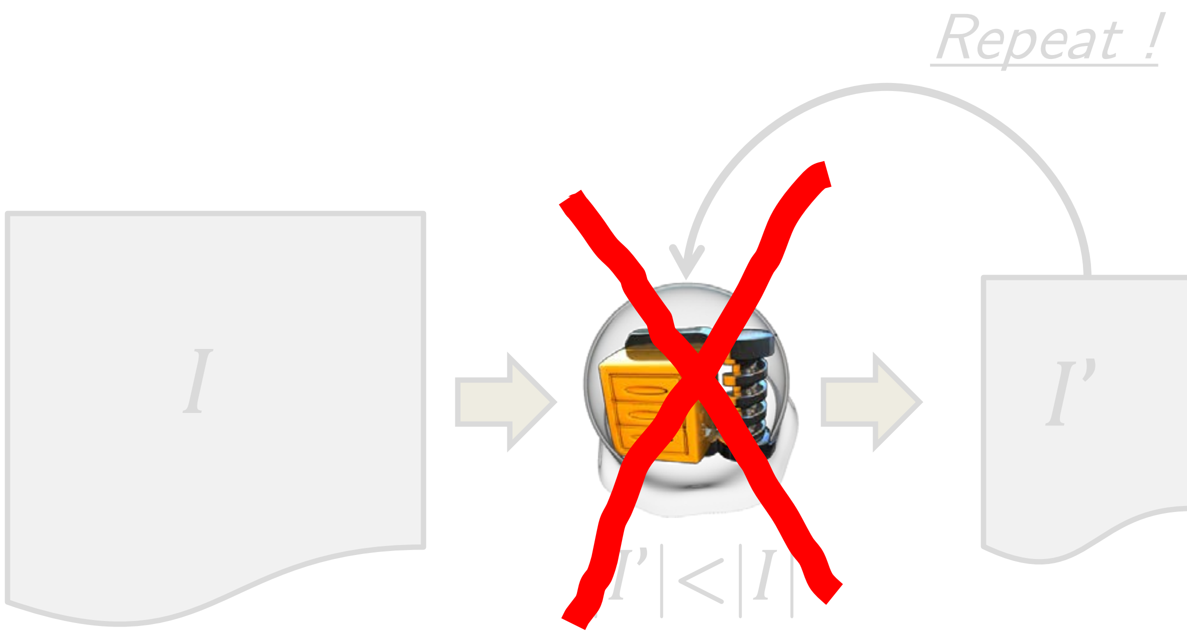
Let us examine "*compression*" in *theoretical computer science*.

Compressing NP-hard problem?



P-time, *preserving membership.*

Compressing NP-hard problem?



Implies $\underline{\mathrm{P} = \mathrm{NP}}$ *!*

Kernelization: Compression in Parameterized Complexity

- *Kernelization* for L is an algorithm:
  - Input: (x, k),
  - Output: (x', k') persereving membership and
    - $|x'| \leq f(k)$ for some function f.
    - k' is bounded by a function of k.

- Many nice upper bounds.

- What about lower bounds?

Machinery for *super-poly* kernel lower bound

- *Composition*: *an important notion.*
  - Input: $(x_1, k), \ldots, (x_t, k)$
  - Output: (y, k')
  - Constraints:
    - Polynomial time computable
    - $y \in L \iff x_i \in L$ for some i
    - $k' \leq \mathrm{poly}(k)$

Theorem [BDFH '08, FS '08]

Assume $\tilde{L}$ is $\mathbf{NP}$-hard. If L *composes* and *has polynomial kernel* then PH *collapses*.

Compose + *Poly-kernel* = PH *collapses*

Polynomial Kernel Lower Bound

- Question: *Polynomial kernel lower bound*?

- [DvM '10] gives some answers:
  - Vertex-Cover has *no* size-$O(k^{2-\varepsilon})$ kernel for any $\varepsilon > 0$.
  - Same for Π-Vertex Deletion.
  - Through *sparsification lower bound*.

- Still, *kernel lower bounds for many* natural problems with *poly kernel* remain open.

Our Contribution

- $\Omega(k^{d-3-\epsilon})$ *kernel lower bound* for:
  - d-Set Packing, d-Set Covering, d-Hitting Set with Bounded Occurrences (through d-BRPC).

- $\Omega(k^{d-4-\epsilon})$ *kernel lower bound for:*
  - d-Clique Packing.

- Extend all known *super-polynomial kernel* lower bounds to *super-quasi-polynomial*.
  - Assume that *exponential time hierarchy* does not collapse.

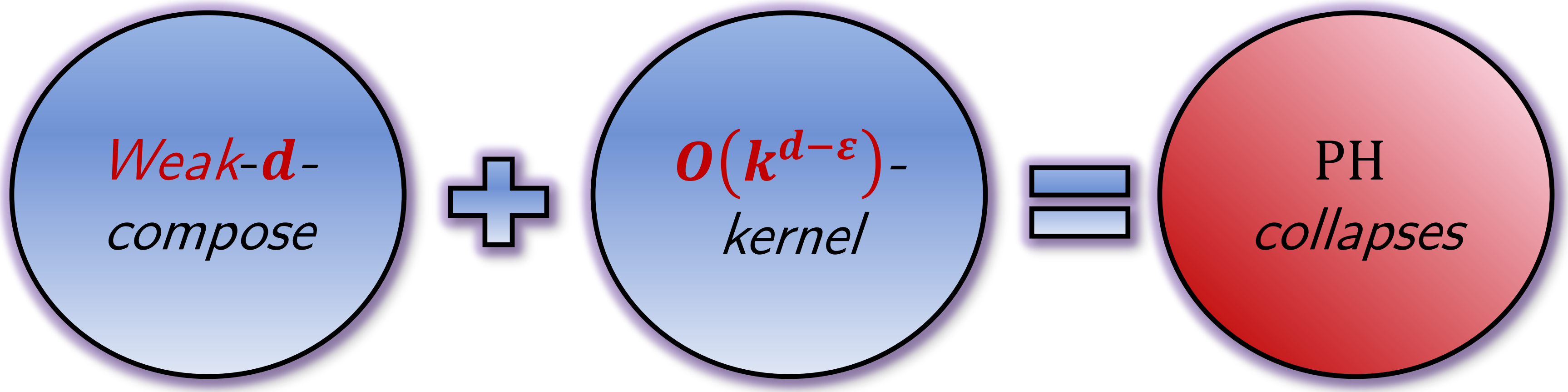
Weak Composition

- Weak d-composition from L_1 to L_2.
  - Input: $(x_1, k), \ldots, (x_t, k)$ of L_1.
  - Output: (y, k') of L_2.
  - Constraints:
    - Polynomial time computable.
    - $y \in L_2 \Leftrightarrow x_i \in L_1$ for some i.
    - $k' \leq t^{1/d} p(k)$ for some fixed polynomial $p(\cdot)$.

Weak Composition

Theorem
Assume $\widetilde{L_1}$ is $\mathbf{NP}$-hard. If there exists a weak d-composition from L_1 to L_2, and L_2 has a *kernel of size* $O\left(k^{d-\varepsilon}\right)$ for some $\varepsilon > 0$, then $\mathbf{PH}$ collapses.

Weak Composition



Main Problem for Lower Bounds

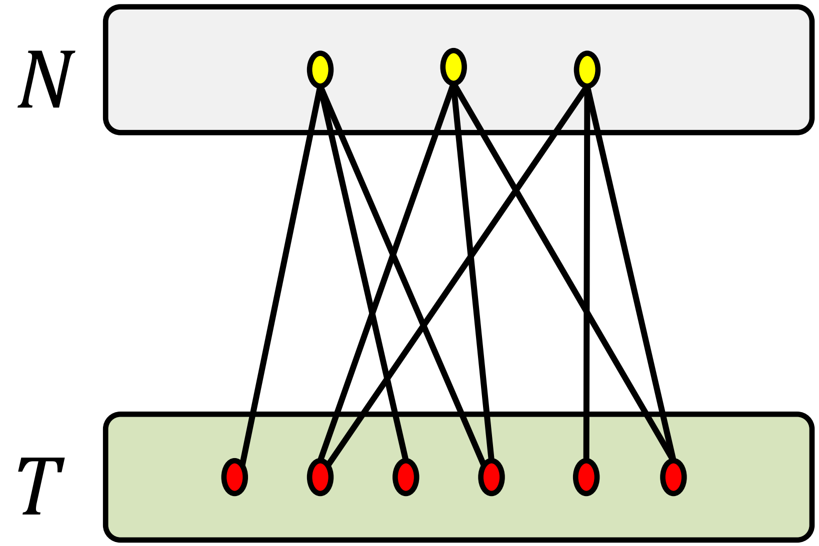
d-BRPC

Instance $\langle G = (N \uplus T, E), k \rangle$,

$(\forall v \in N)\ \deg(v) = d$.

Membership $(\exists N' \subseteq N, |N'| = k)$

$(\forall v \in T)(|\Gamma(v, N')| = 1)$

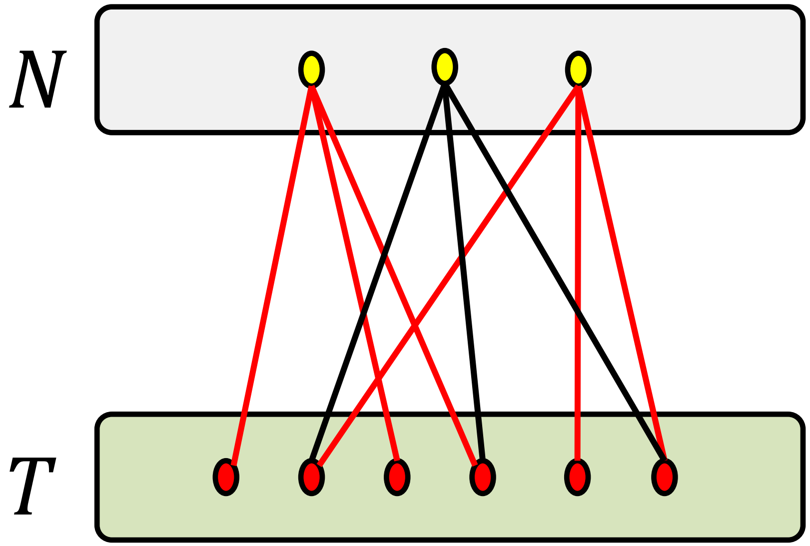


Main Problem for Lower Bounds

d-BRPC

Instance $\langle G = (N \uplus T, E), k \rangle$,
$(\forall v \in N)\ \deg(v) = d$.

Membership $(\exists N' \subseteq N, |N'| = k)$
$(\forall v \in T)(|\Gamma(v, N')| = 1)$



Trivial kernel:
removing duplicated non-terminals gives a kernel of size $\binom{kd}{d} = O(k^d)$.

Main Lower Bound Result

(Our result)
Unless **PH** collapses, d-BRPC has no kernel of size $O\left(k^{d-3-\varepsilon}\right)$, for any $\varepsilon > 0$.

Proof of the main result

Weak Composition for d-BRPC

- *Basic idea*: *colors* and *IDs* [*DLS, ICALP '09*]

- Source problem:
  col-3-BRPC
  - Instance: BRPC instance, with a color mapping $col: N \rightarrow \{1, \dots, k\}$.
  - Question: same question and additionally, N' consists of vertices of different colors?

- Target problem: $(d + 3)$-BRPC.

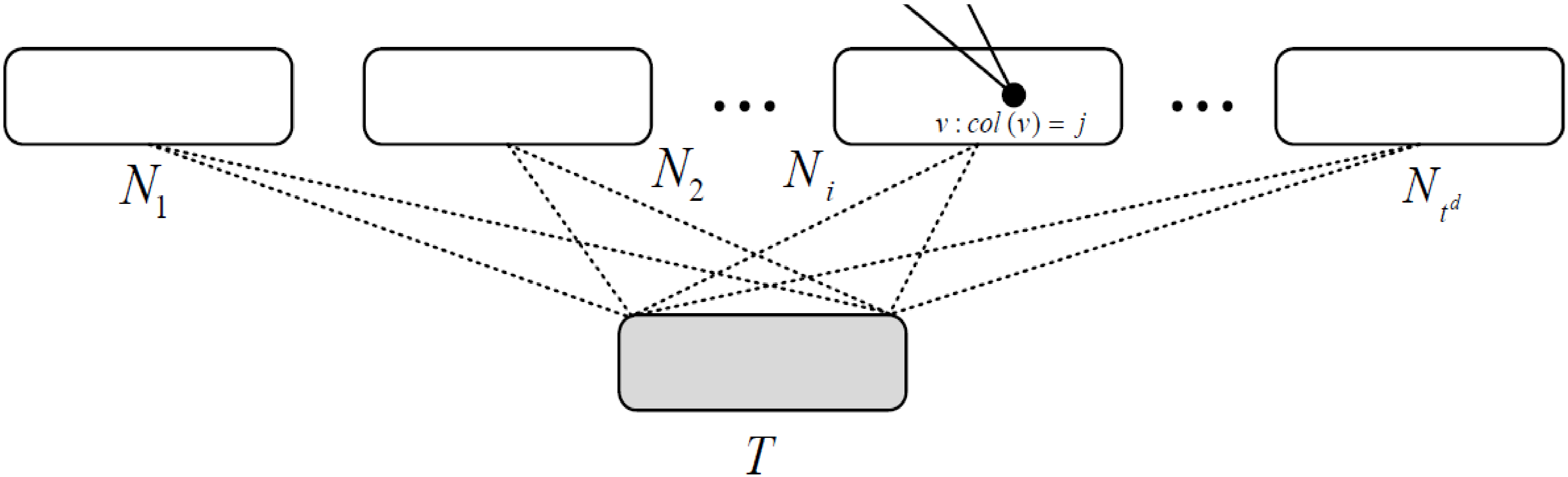
- Compose $\Theta(t^d)$ source instances to target.

Composition: General Setup

- Given input sequence $(G_1, col_1, k), \dots, (G_n, col_n, k)$
  - Assign ID_i to instance N_i, $ID_i \subseteq \{1, \dots (t+d)\}$, $|ID_i| = d$
  - Observe $\binom{t+d}{d} = \Theta(t^d)$

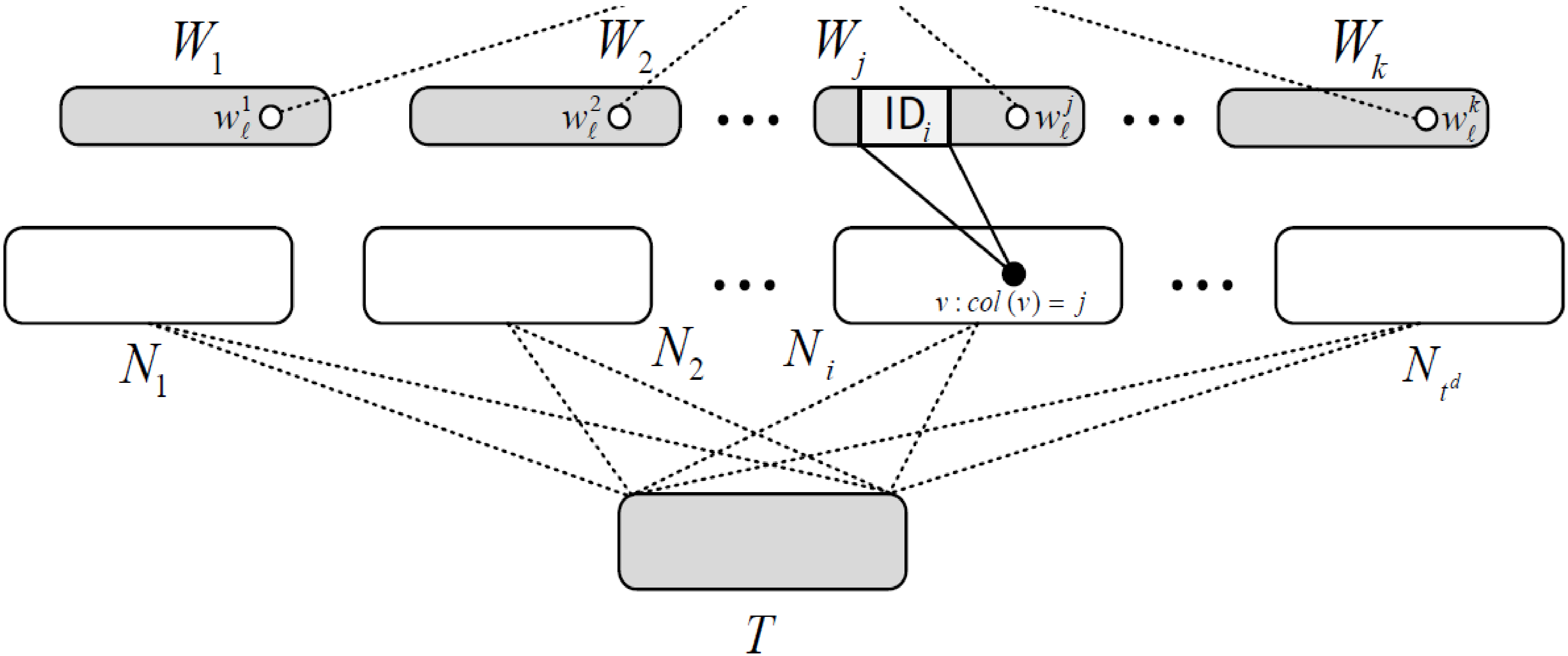
- Overview of composition
  - *First step*: compose to BPC (some vertices have unbounded degree).
  - *Second step*: construct equality gadget to get regular degree instance.

BRPC to BPC (1/3)



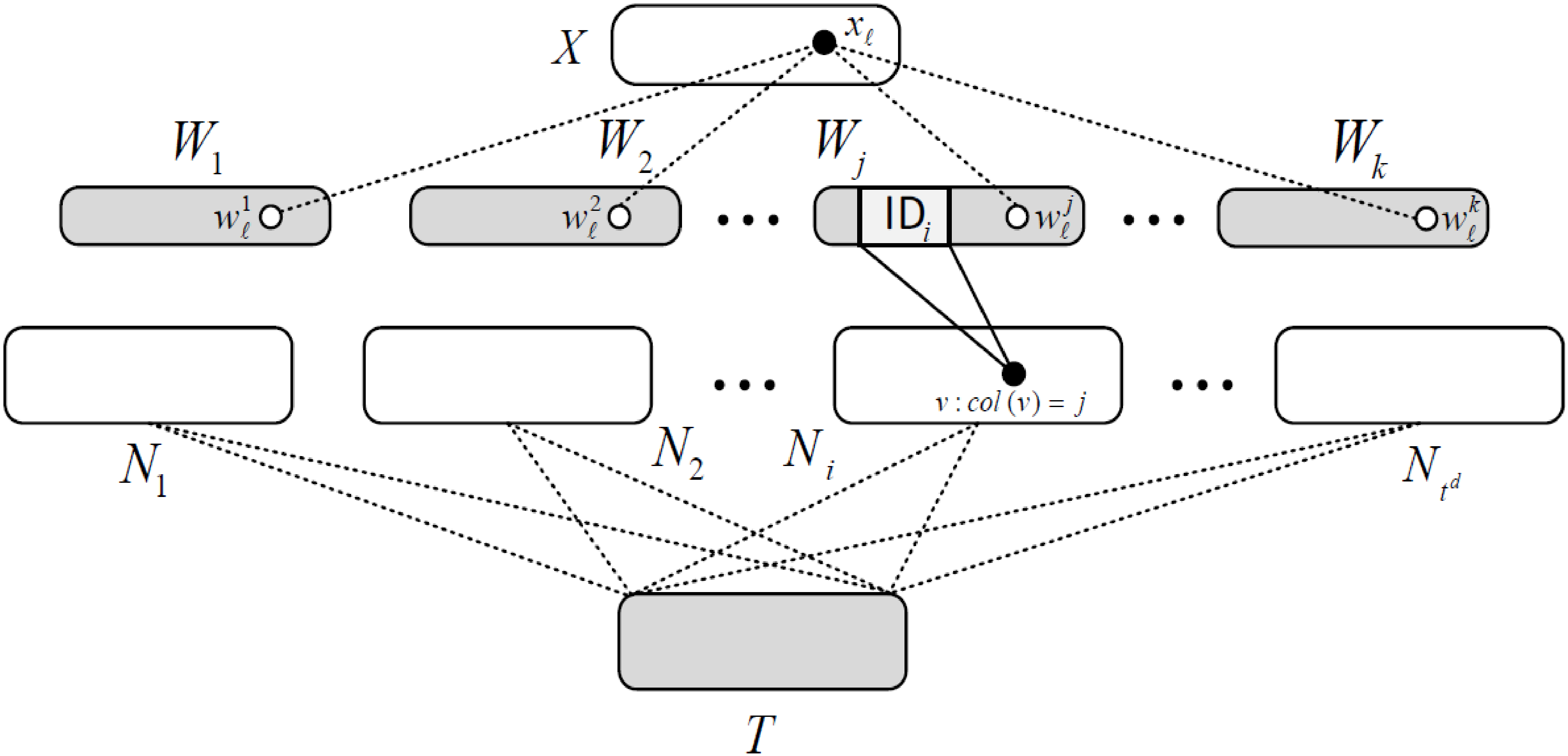
Set $k' = k + t$

BRPC to BPC (2/3)



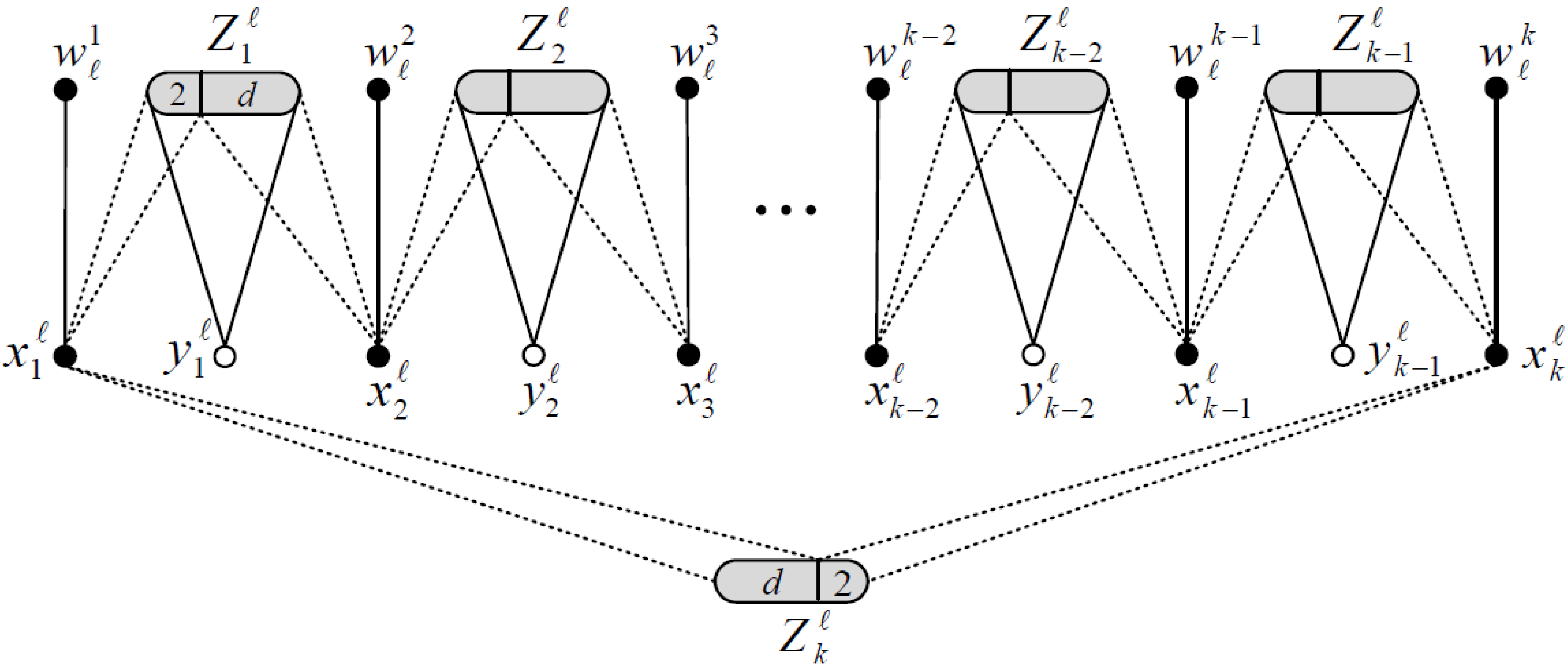
Set $k' = k + t$

BRPC to BPC (3/3)



Set $k' = k + t$

Second Step: Equality Gadgets



- If x_i^ℓ is chosen, then forced to choose all x_i^ℓ's.
- *Otherwise* all y_i^ℓ's must be picked.

Wrap Up

- *Lower bound* for d-BRPC.

- By *reduction*, get lower bounds for others.
  - *Linear parameter transformation.*

Super-Quasi-Polynomial Kernel

- Adapt Fortnow-Santhanam-Dell-van Melkebeek argument to *quasi-polynomial* case.
- *Implication*: quasi-polynomial kernel implies *collapse of exponential time hierarchy*.
  - Use an analog of Yap's theorem in exponential hierarchy.
- *Observation*: previous super-polynomial lower bounds.
  - Via Composition
  - Via Polynomial parametric transformation.

Open Problems

- Close the *gap* between upper and lower bound?
  - We have seen some of them some time before.
- Lower bounds for *more* problems?
  - Got bounds for *matching* problems.
- Exclude *subexponential kernels*?
  - All known techniques seem to cease to work.
- …

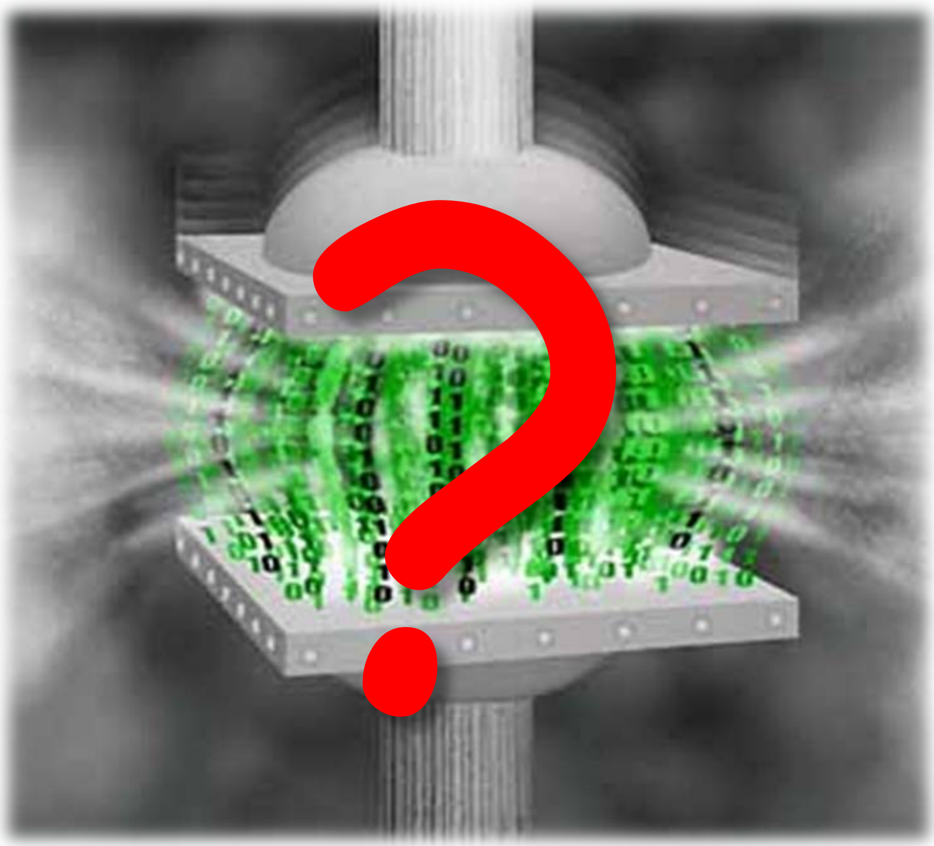
Conclusion

- *A formulation of weak composition*
  - for proving *polynomial lower bounds* for kernelization.

- *Polynomial kernelization lower bounds*
  - for some natural parameterized problems.

- *Super-quasi-polynomial kernel lower bounds*

Thanks !



(And if you really want to know more about history...)

Backup Slides

Compressibility of $\mathbf{NP}$ instances

- [Harnik, Naor, FOCS '06, SICOMP '10]
  - Let L be a language in $\mathbf{NP}$,
  - $n :=$ instance size, $k :=$ witness size.

  - A *errorless compression* is an algorithm $\boldsymbol{A}$ with language L' and a polynomial $p(.,.)$:
    - Size of $\boldsymbol{A}(x)$ is at most $p(k, \log n)$.
    - $\boldsymbol{A}(x)$ in L' iff x in L.

Compressing $OR(L)$

- $OR(L)$
  - L in **NP**.
  - Input: instances $x_1, \dots, x_t$ each of length n.
  - Membership: $(x_1, \dots, x_t)$ in $OR(L)$ if there exists i in $[t]$ such that x_i in L.

- Observation: the *witness size* is n, the *size of the instance sequence* is dominated by t.

Compressing $OR(L)$

- $OR(L)$

Question 1

Does *OR(SAT)* has a *compression algorithm*?

(wait…. why is this interesting at all?)

- Observation: the *witness size* is n, the *size of the instance sequence* is dominated by t.

On the *Positive* Side...

Theorem [Harnik, Naor, FOCS '06]

If *errorless compression* for $OR(SAT)$ exists, then can construct a family of *collision-resistant hash functions* (CRH) on *any one-way function* (OWF).

- No known construction of CRH from general OWF.
- Impossible with OWP using *black-box reductions*.

On the *Negative* Side...

The *incompressibility of $OR(SAT)$* allows

Investigation of *incompressibility* of

other interesting problems.

(Not necessarily $OR(L)$!)

Question 2

How? Coming soon...

Compressing $OR(SAT)$ is unlikely

- *Distillation* for $OR(SAT)$:
  - For some fixed polynomial $p(\cdot)$;
  - Input: $\phi_1, \ldots, \phi_t$, each of length n. $t = \text{poly}(n)$
  - Output: $\phi \in \text{SAT} \Leftrightarrow \phi_i \in \text{SAT}$ for some i.
  - Constraints:
    - Runs in time polynomial in length of input sequence.
    - $|\phi| = p(n)$; independent of t.

- Distillation is just a *special case* of compression

Compressing $OR(SAT)$ is unlikely

- *Distillation* for $OR(SAT)$:
  - For some fixed polynomial $p(\cdot)$:

Theorem [Fortnow et al, STOC '08]

If there is *distillation algorithm* for $OR(SAT)$, then $\mathrm{coNP} \subseteq \mathrm{NP/poly}$: polynomial hierarchy collapses.

- Distillation is just a *special case* of compression

Connect to Parameterized Complexity

- In the compression of size $p(k, \log n)$, if dropping the dependence on $\log n$. Then,

- The question is equivalent to find a kernel of size $p(k)$ in parameterized complexity.
  - Where the parameter is the witness size.

- Now turn into *parameterized complexity*.