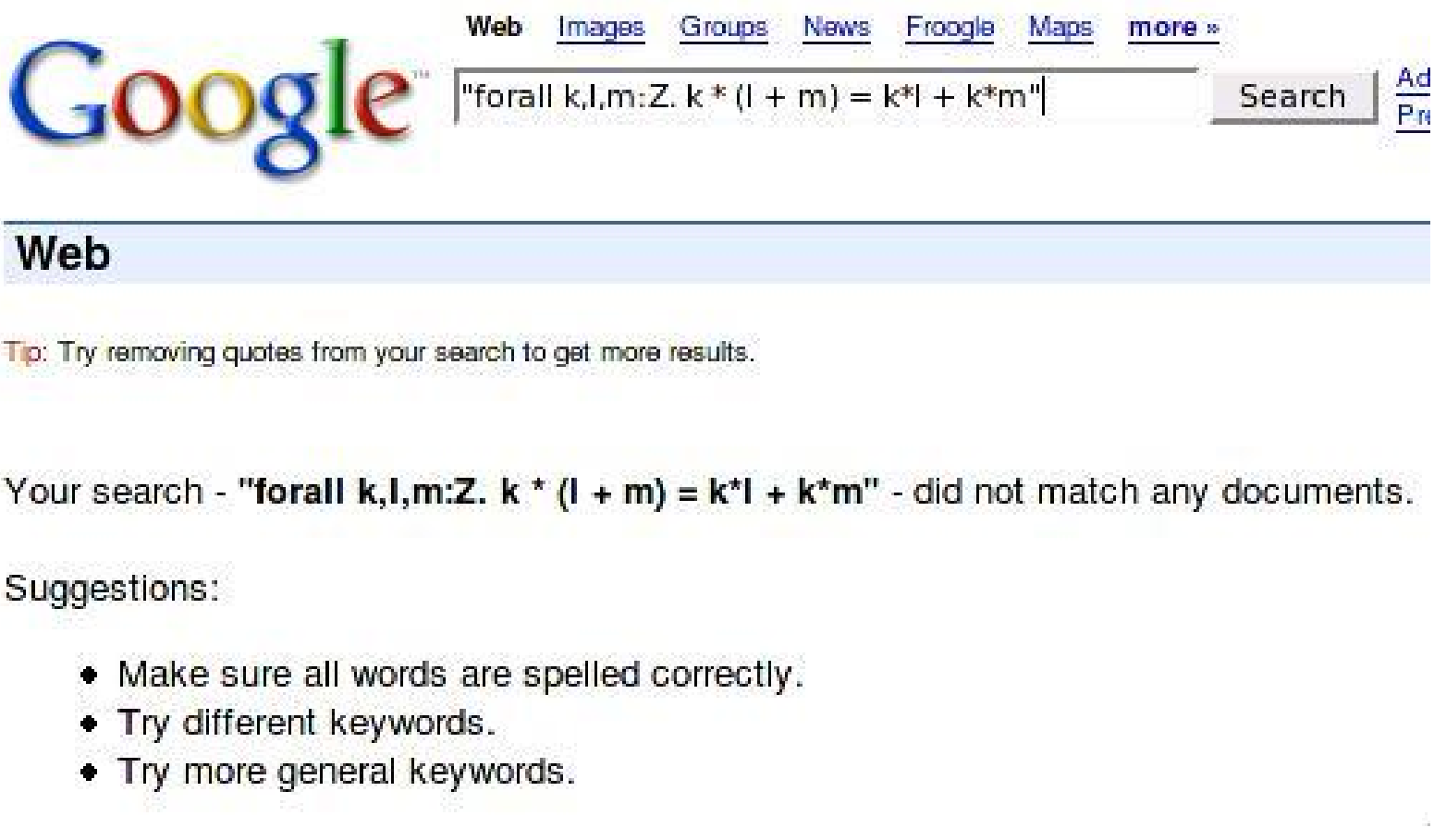
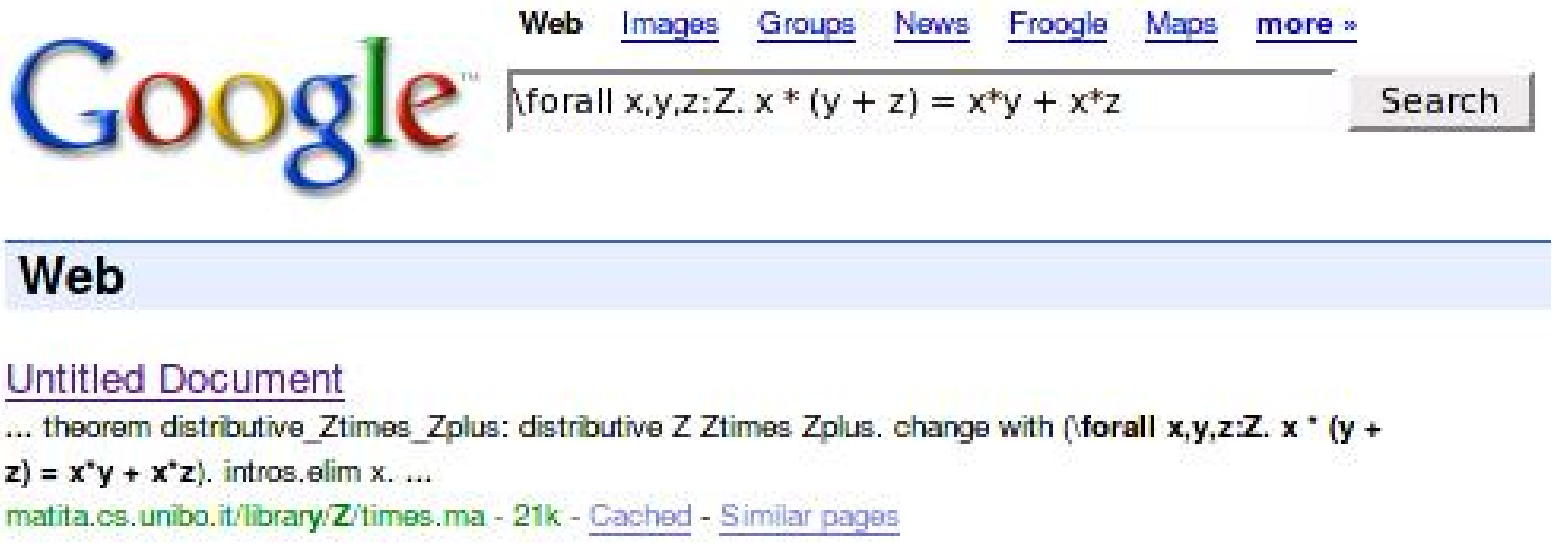
Searching for Distributivity



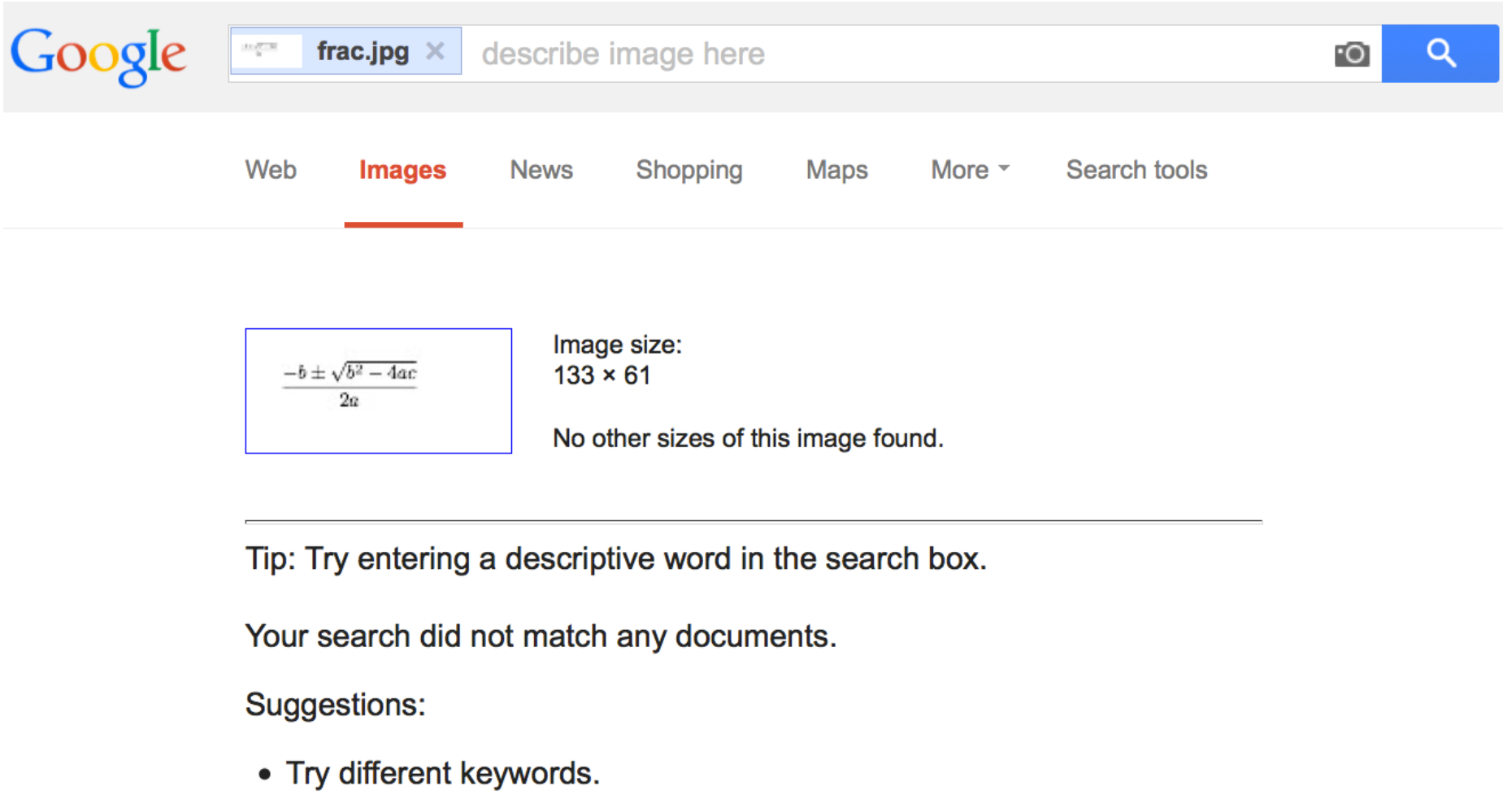
Searching for Distributivity



Searching for Distributivity

Does Image Search help?

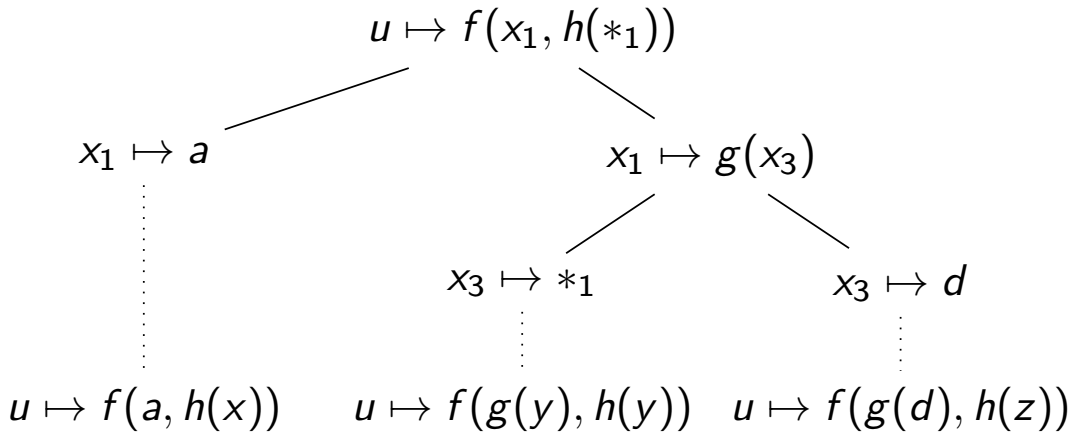
- Math formulae are visual objects, after all (let's try it)



Of course Google cannot work out of the box

- Formulae are not words:
  - a, b, c, k, l, m, x, y, and z are (bound) variables.(do not behave like words/symbols)
  - where are the word boundaries for "bag-of-words" methods?
- Formulae are not images either: They have internal (recursive) structure and compositional meaning
- Idea: Need a special treatment for formulae (translate into "special words") Indeed this is done ([**MilYou:tadlmf02**; **MunMin:MathFind06**; **LibMel:marmca06**; **MisGal:egoMath11**]) . . . and works surprisingly well (using e.g. Lucene as an indexing engine)
- Idea: Use database techniques (extract metadata and index it) Indeed this is done for the `Coq/HELM` corpus ([**AGSTZ:ContMathSearchWhelp04**])
- Our Idea: Use Automated Reasoning Techniques (free term indexing from theorem prover jails)
- Demo: MathWebSearch on Zentralblatt Math, the arXiv Data Set

Substitution Tree [Graf '94]

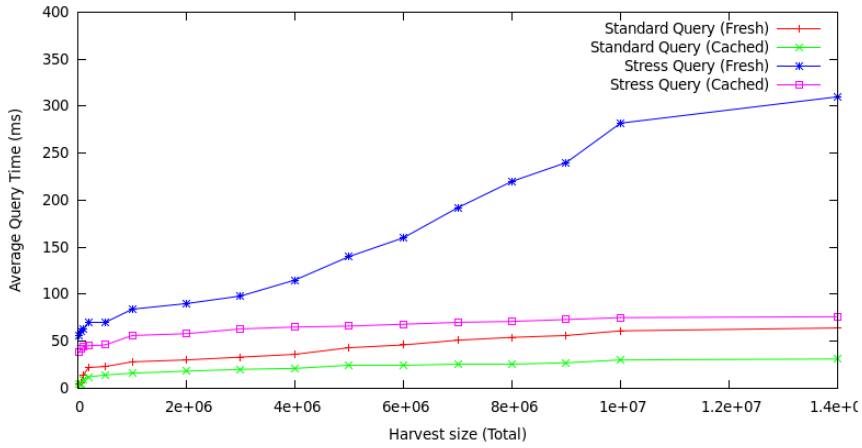


- Variant of abstraction trees that indexes Substitutions(Nodes labeled with Substitutions)
- includes Variable renaming ($*_i \widehat{=} i^{\text{th}}$ variable)
- less redundant than abstraction trees
- allows $n : m$ indexing

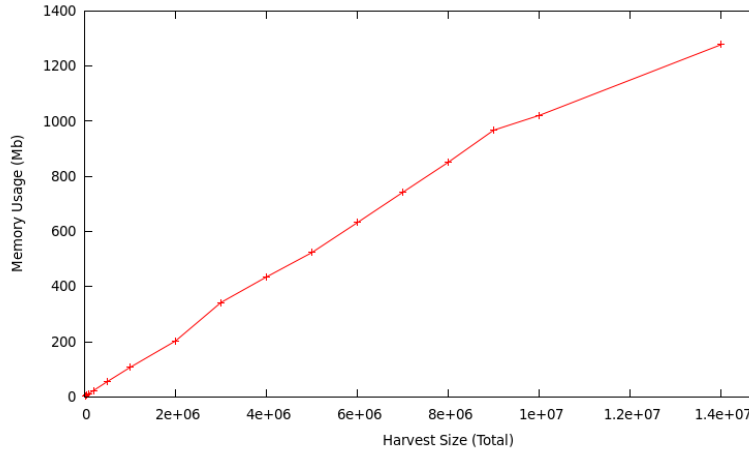
Index statistics

- Experiment: Indexing the arXiv (1M documents, $\sim 10^8$ non-trivial formulae)
- Results: indexing up to 15 M formulae on a standard laptop

Query Times



Memory Footprint



  - query time is constant ($\sim$ 15ms) (as expected; goes by depth $\times$ symbols)
  - memory footprint seems linear ($\sim 500 \frac{\text{B}}{\text{formula}}$) (expected more duplicates)
- So we need ca. 100G B RAM for indexing the whole arXiv.
- Can index all published Math ($\widehat{=}$ 5 $\times$ arXiv) on a large server (.5T B RAM). (ZBL $\widehat{=}$ 3.5M art.)

Formula/Text Search Combination?

- Observation: `MathWebSearch` is similar to a one-word IR algorithm, except . . . unification directly matches one search term against lots of search terms.
- Idea: combine unification indexing with the vector space model for a "bag-of-formulae" (instead of standard IR's "bag-of-words") method . . .
- at Indexing time: when we index a math document D,
  - insert the formulae into the `MathWebSearch` index (remember dbid)
  - replace all formulae in D with their dbid to get D'
  - index D' in a bag-of-words index (e.g. Elastic Search or Terrier)
- At query time: (essentially query expansion)
  - query Q consists of a set Q_f of formulae and a set Q_w of words.
  - run Q_f through `MathWebSearch` to get set I_f of matching dbids.
  - run $Q' = Q_w + I_f$ through nutch to get a set R of document fragments URIs.
- we return R together with the fragments of D they point to.
- we can even inherit the ranking mechanisms from nutch. (see if they help)