Finite Model Finding for SMT

Andrew Reynolds
University of Iowa

April 26, 2012

Acknowledgments

- Intel Corporation
- University of Iowa
- New York University

CVC4: SMT Solver

- SMT Solver
- Support for many theories
  - Equality + Uninterpreted Functions
  - Integer/Real arithmetic
  - Bit Vectors, Arrays, Datatypes
- Other features: Proofs
- Work in progress: Quantifiers
  - Pattern-based instantiation
  - Model-based instantiation
  - Rewrite Rules
  - *Finite Model Finding*

Quantifiers in SMT

- SMT solvers
  - Powerful tools for determining satisfiability of ground formulas
    - DPLL(T) for finding SAT assignments to ground formulas
    - Answer UNSAT if no model can be found
  - However, difficult to answer SAT in the presence of universal quantifiers

Quantifiers in SMT

- Given set of literals (G, F):
  - Set of ground constraints G
  - Set of quantified assertions F
- Questions:
  - (1) How to chose instantiations for F
  - (2) When can we answer SAT?

Other approaches

- Pattern-Based Instantiation
  - Determine instantiations heuristically
    - Based on finding ground terms in G with same shape as terms in F
  - Usually these methods cannot answer SAT
- Complete Instantiation
  - Determine sufficient set F^* of instantiations
  - If F^* is satisfiable, we know F is satisfiable
    - Only applicable to some fragments of first-order logic
- Model-Based Instantiation
  - Determine instantiations based on possible counterexamples (from F) to current model for G
  - Can answer SAT if counterexamples are proven impossible

Finite Model Finding

- Finite Model Finding (for EUF)
  - Find smallest model for ground constraints
    - Instantiate exhaustively with terms in this model
  - Answer SAT if exhaustive instantiation is consistent with model
    - Practical if small models exist
    - Can extend to quantifiers over finite sorts
      - » Finite Datatypes, BitVectors, …

Finite Model Finding: Overview

- Wish to find reasonably small models
  - Impose *cardinality constraints* on (uninterpreted) sorts
  - Try models of size 1, 2, 3, … etc.
- What this requires:
  - Control to DPLL(T) search for postulating cardinalities
  - Solver for UF+cardinality constraints
  - Strategy for instantiating quantifiers exhaustively
    - May reduce # instantiations
      - Only try instantiations that are relevant to the model

UF+Cardinality Constraints

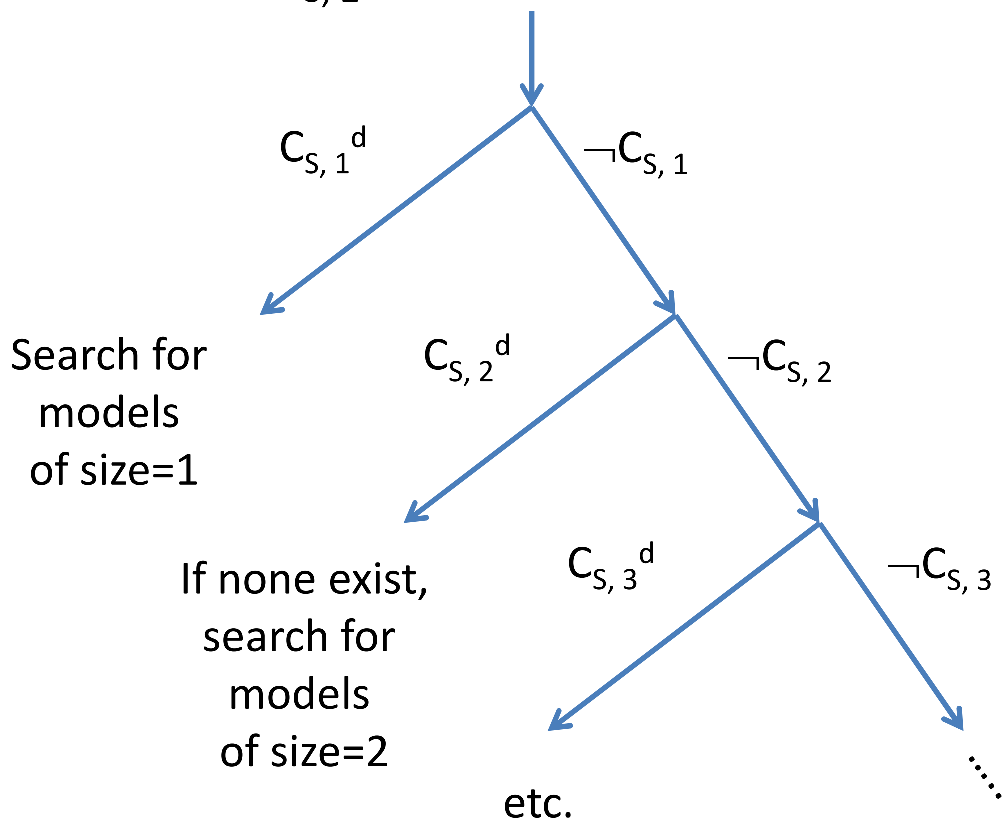
- Extend UF to handle literals of the form:

$$C_{S,\ k}$$

- Meaning "the cardinality of sort S is less than or equal to (integer) k"
  - i.e., at most *k* equivalence classes of sort *S* exist

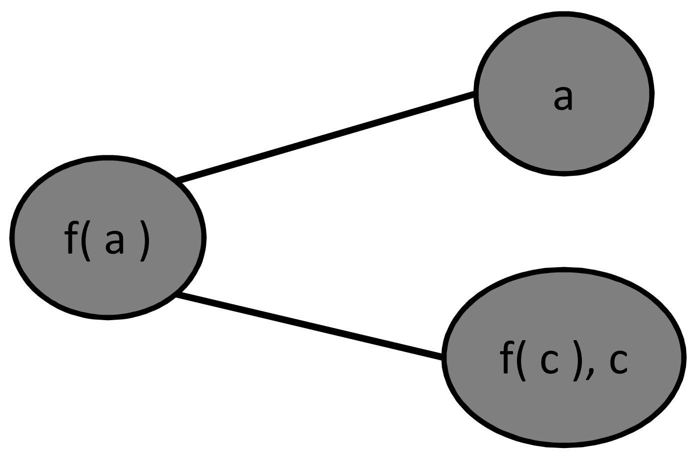
DPLL(T) for UF+Cardinality

- Idea: try to find models of size 1, 2, 3…etc.
  - Choose $C_{S,1}^d$ as first decision literal
  - If fail, then try $C_{S,2}^d$, etc.



UF+Cardinality Constraints

- For each sort S, maintain disequality graph $D_S = (V, E)$
  - V are equivalence classes of sort S
  - E are disequalities between terms of sort S
- D_S induced by asserted set of literals
  - So, $f(a) \neq a$, $f(a) \neq c$, $f(c) = c$ becomes:

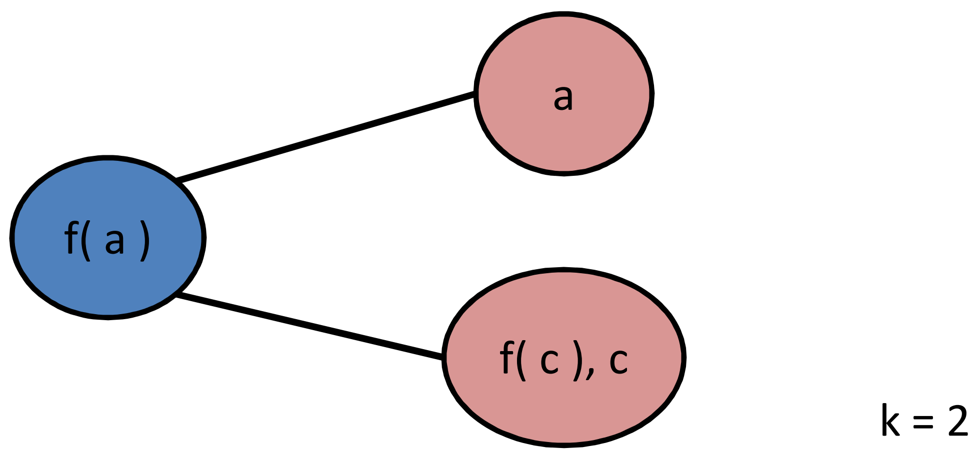


UF+Cardinality Constraints

- Must extend theory solver for UF
  - Determine when no models of size k exist
  - If benchmark contains no function symbols
    - Can use k-colorability algorithm
  - More difficult with function symbols
  - In either case, problem is NP-hard

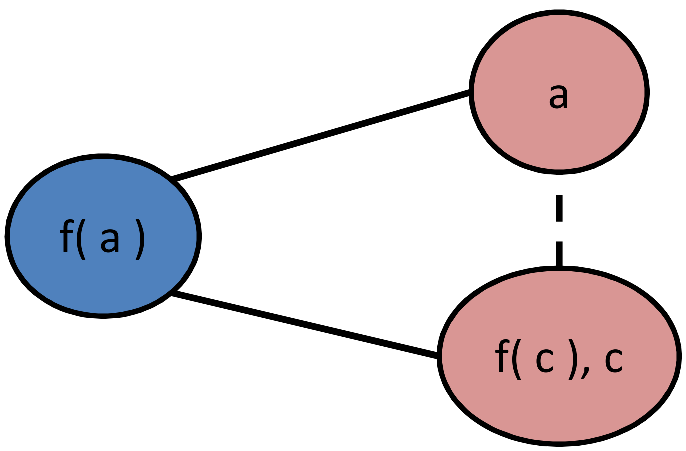
UF+Cardinality Constraints

- Assume a single sort S with cardinality constraint k
  - We are interested in whether D_S is k-colorable
    - If *no*, then we have a conflict ($\psi \Rightarrow \neg C_{S,k}$)
      - where ψ is explanation of sub-graph of D_S that is not k-colorable
    - If *yes*, then we *cannot* be sure a model of size k exists
      - Identifying elements may have consequences for theories
      - Example: congruence axioms in UF



UF+Cardinality Constraints

- Solution: must explicitly shrink model
- Use splitting on demand
  - Add lemma (a = f(c) $\vee$ a $\neq$ f(c))
  - Explore the branch a = f(c) first
    - If successful,
      - We shrink # of equivalence classes by one
    - If unsuccessful,
      - A theory conflict/backtrack will occur
        - » May or may not involve cardinality constraints



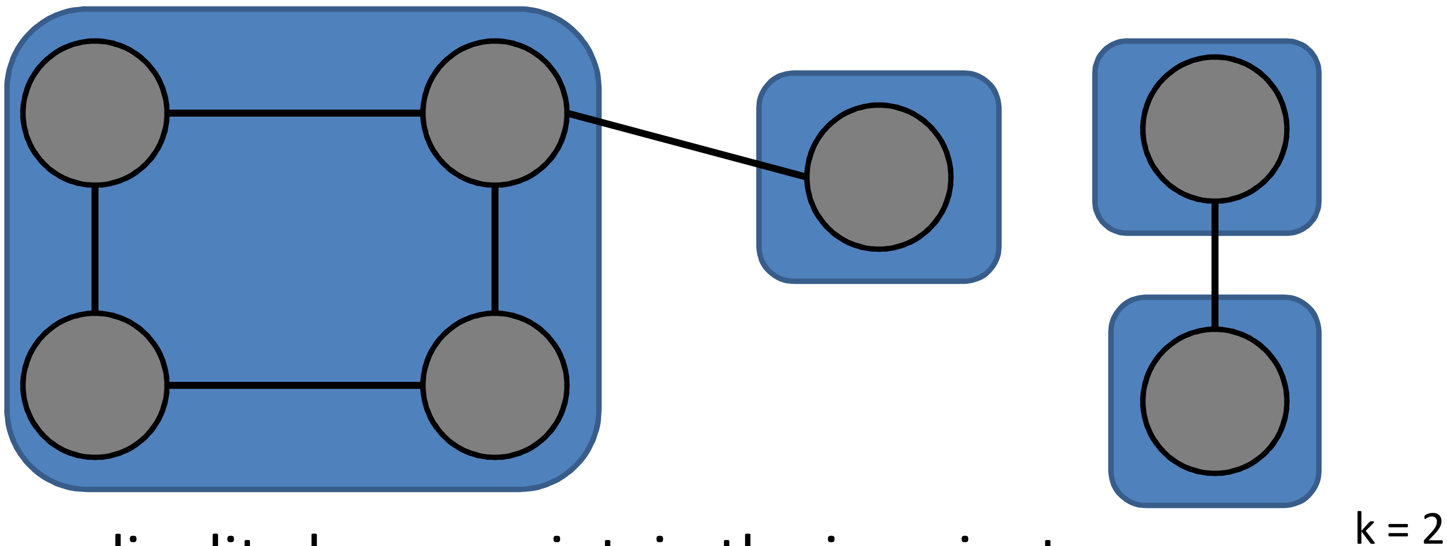
k = 2

UF+Cardinality Constraints

- Strategy for UF+Cardinality must be:
  - Able to recognize when D_S is not k-colorable
  - Helpful for suggesting relevant splits
- Solution: use a *region-based approach*
  - Partition nodes in *regions* with high edge density
    - Likely to find cliques
    - Can suggest relevant splits

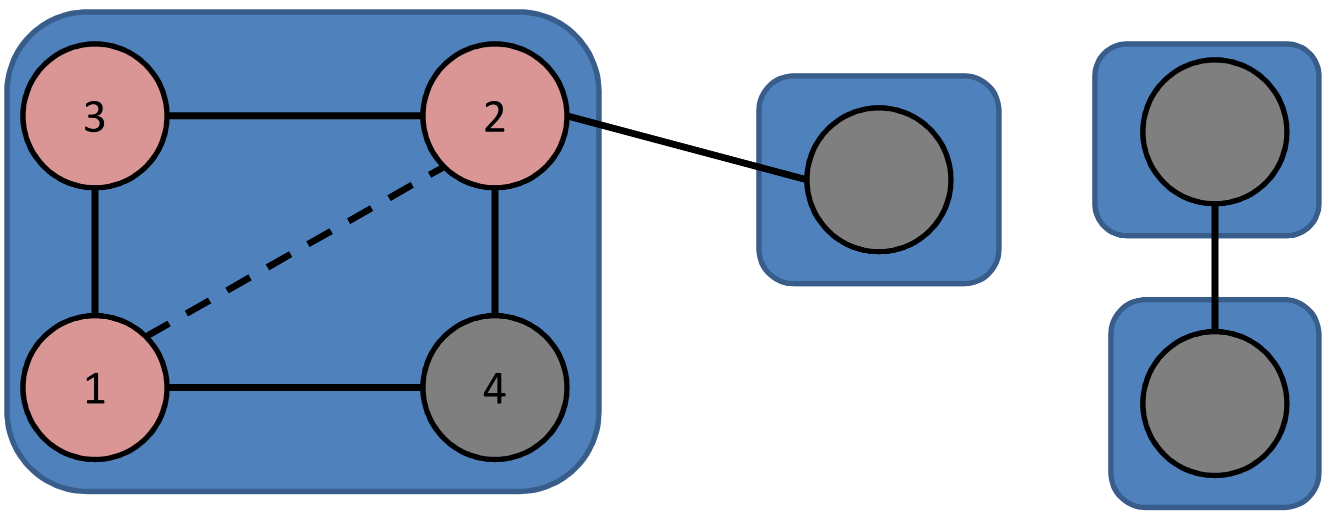
Region-Based Approach

- Partition nodes V of D_S into *regions*



- For cardinality k, we maintain the invariant:
  - No clique of size k+1 exists containing nodes from multiple regions
- Thus, we only need to search for cliques local to regions
  - Region can be ignored if it has ≤ k terms

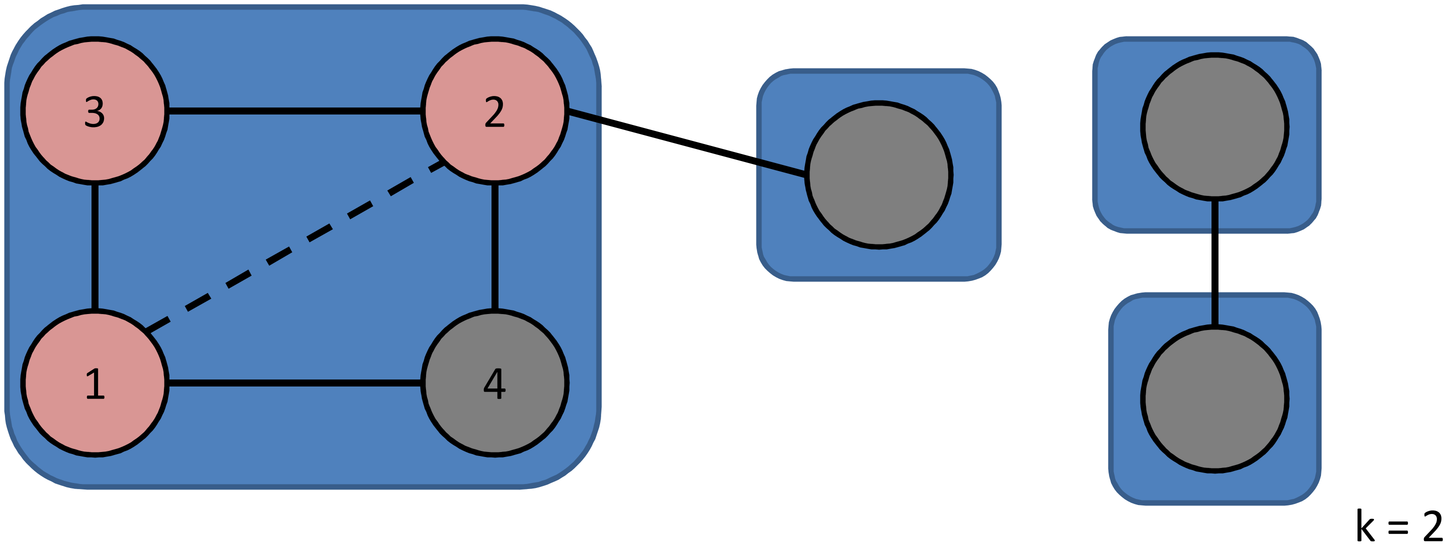
Region-Based Approach



k = 2

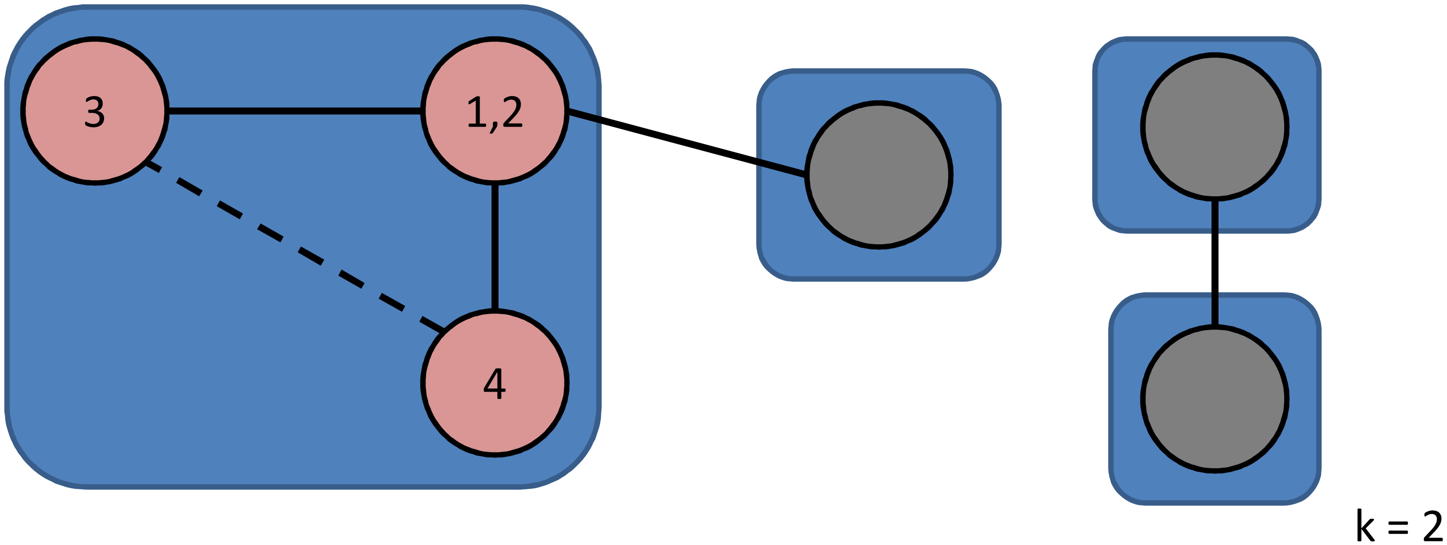
- Within each region with size > k:
  - Maintain a watched set N of k+1 nodes
  - Record pairs of nodes in N that are not linked
    - If this set is empty, N is a clique $\Rightarrow$ report a conflict clause
    - Otherwise, guess equalities over unlinked nodes in N

Region-Based Approach



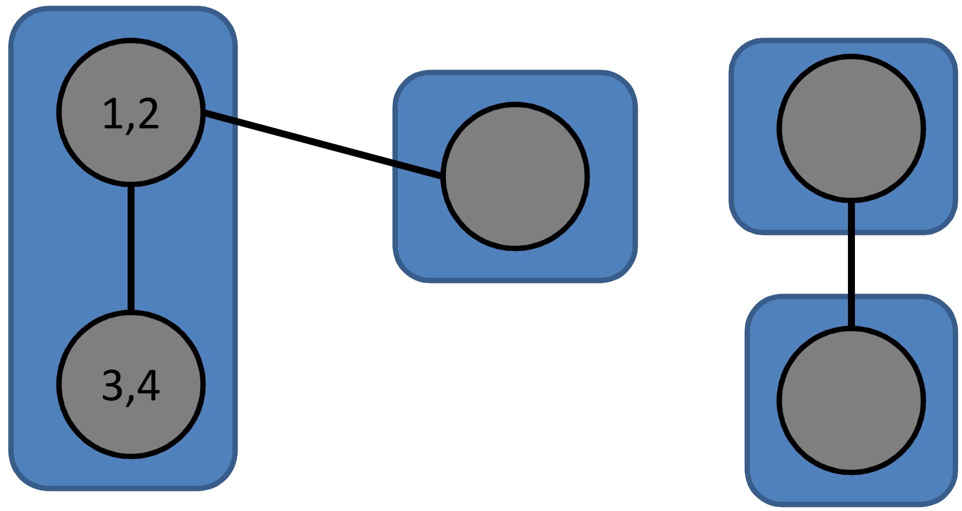
- Merging nodes 1 and 2 may:
  - Lead to a theory conflict
  - Lead to a cardinality conflict (force a clique), or
  - *Succeed*

Region-Based Approach



- When merge is successful,
  - Continue guessing equalities until all regions have ≤ k nodes

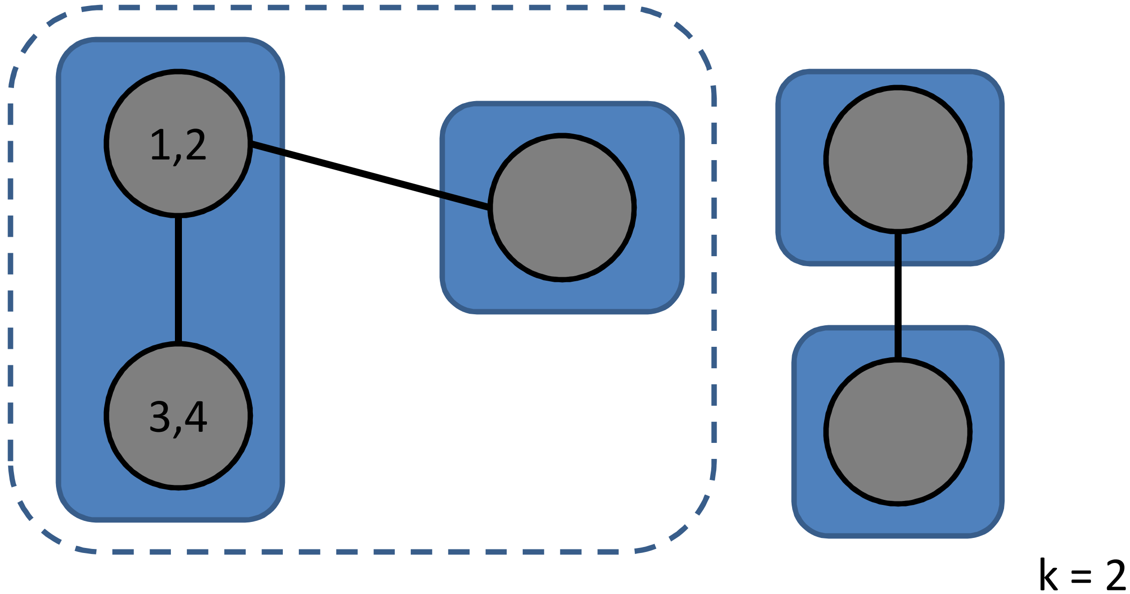
Region-Based Approach



k = 2

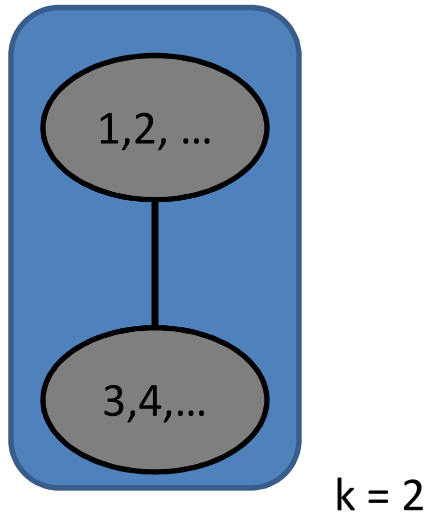
- All regions have ≤ k nodes
  - At this point, we are ensured k-colorability
  - However, still unsure a model of size k exists
    - Again, due to possible theory conflicts
  - *Must shrink model explicitly*

Region-Based Approach



- Combine regions based on heuristics
  - For example, # edges between regions

Region-Based Approach



- Continue combining regions, guessing equalities until we have until ≤ k nodes overall
  - When this is the case, we have model of size k for S

UF+Cardinality Constraints Summary

- For cardinality k, maintain a partition into regions
  - At *weak* effort check,
    - If any cliques of size k+1 exist:
      - report them as conflicts clauses
  - At *strong* effort check,
    - If # representatives for sort S ≤ k:
      - return SAT
    - Otherwise, if there is any region R, $|R| > k$:
      - add splitting lemma between terms within R
    - Otherwise:
      - combine regions, repeat strong effort check
- Both checks can be performed quickly

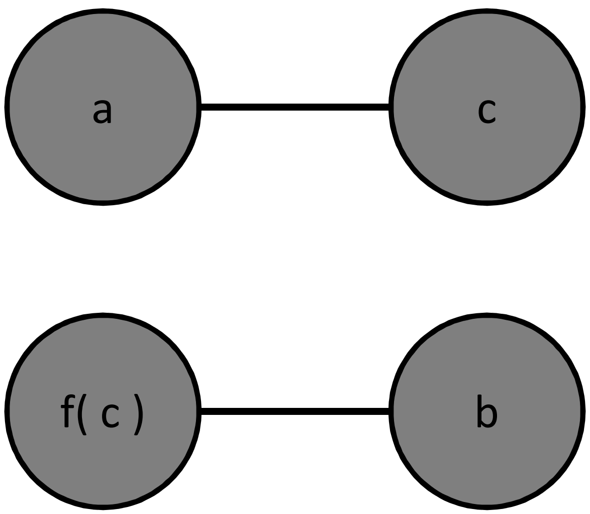
Finite Model Finding

- Use DPLL(T) to guide search for small models
  - Use solver for UF+cardinality constraints
- Why small models?
  - Easier to test against quantifiers
    - Assuming model is small,
      - Instantiate quantifiers w all combinations of representatives
      - If we have same model after instantiation,
        - » Model satisfies quantifiers, able to answer SAT

Instantiation: Example 1

- Assertions:

$$a \neq c,\ f(c) \neq b,\ \forall xy.\ f(x) \neq g(y)$$

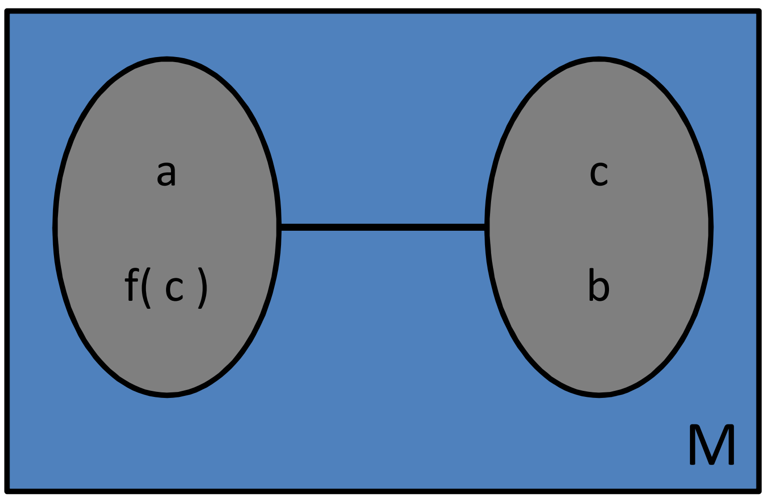


Instantiation: Example 1

- Assertions:

  $a \neq c,\ f(\ c\ ) \neq b,\ \forall xy.\ f(\ x\ ) \neq g(\ y\ )$

- Find minimal model M, cardinality 2:

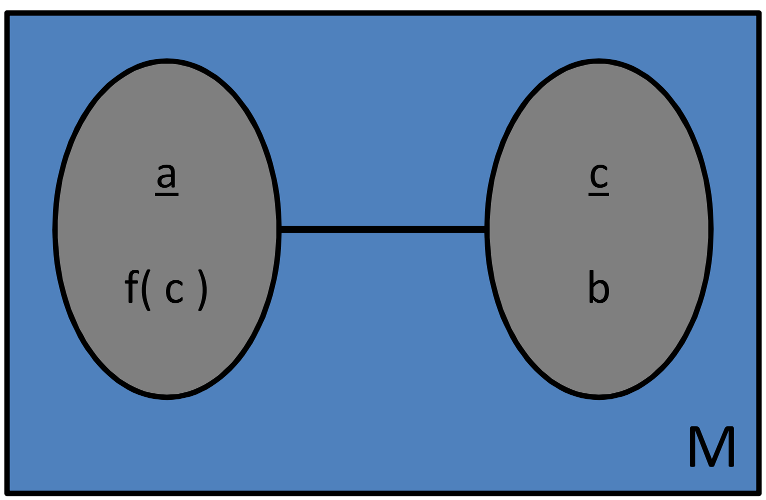


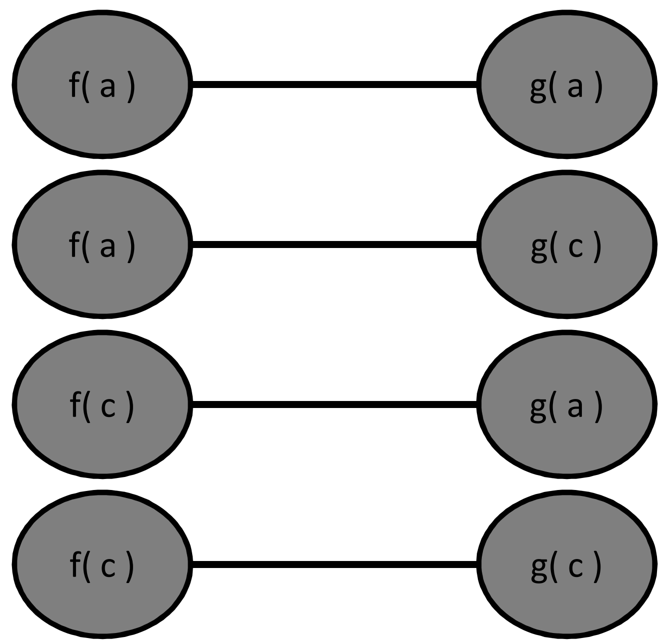
Instantiation: Example 1

- Assertions:

$$a \neq c,\ f(\ c\ ) \neq b,\ \forall xy.\ f(\ x\ ) \neq g(\ y\ )$$

- Instantiate quantified formula with reps a, c:



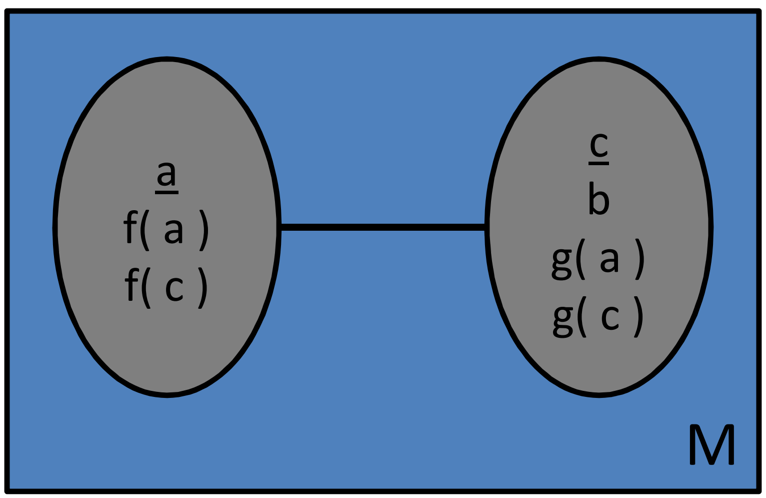


Instantiation: Example 1

- Assertions:

  $a \neq c$, $f(c) \neq b$, $\forall xy.\ f(x) \neq g(y)$

- Reapply UF+cardinality solver:



- Success:

  M satisfies $\forall xy.\ f(x) \neq g(y)$

- Answer SAT

Possible Improvements

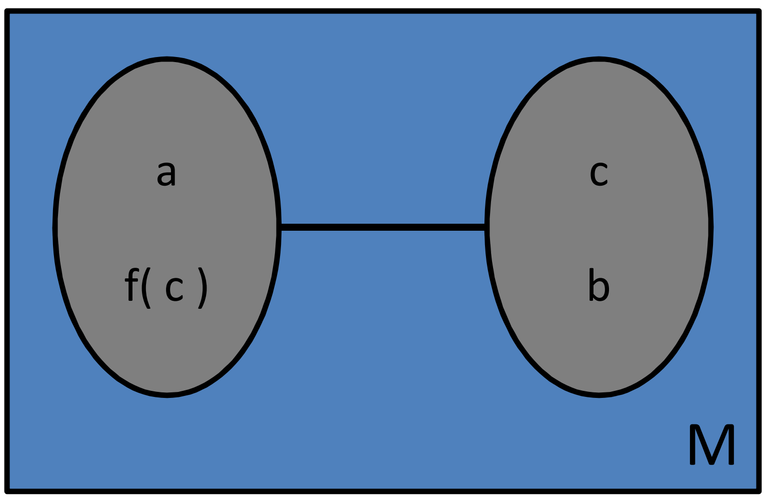
- Exhaustive instantiation
  - Instantiate quantifiers F with *all* combinations of representatives
- Advantages:
  - If successful, we are ensured that F is satisfied by M
- Disadvantages:
  - Produces many instantiations
  - Even small models may cause many instantiations
    - Quantifiers over n variables, # instantiations is $O(k^n)$

- Improvement: Determine tight over-approximation of relevant instantiations to test

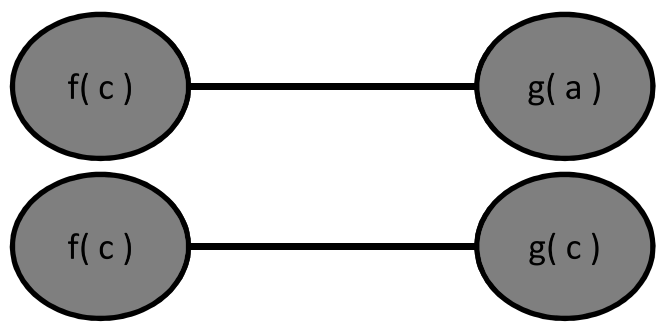
Instantiation: Improvements

- Example 1 revised:

$$a \neq c, f(c) \neq b, \neg P(a), \forall xy. (P(x) \Rightarrow f(x) \neq g(y))$$



- Since $\neg$P(a), our set is:
  x -> { c },
  y -> { a, c }



Instantiation: Improvements

- Possible approaches:
  - Compute over-approximation of relevant instantiations
    - Complete the candidate model M
      - Give interpretation to predicates and functions
      - Define default values heuristically
    - Do not consider instantiations that are already true in the model
      - P(x) in the formula $\forall$xy. (P(x) $\Rightarrow$f(x) $\neq$ g(y))
        - » Do not consider { x $\rightarrow$ a } if $\neg$P(a)
    - *Advantage:* may be fast to compute, reduces # inst
  - Compute exact set of relevant instantiations
    - Complete the candidate model M
    - Use model-based quantifier instantiation
    - Try values for which the negation of the body of quantifier is satisfied
    - *Advantage:* only try instantiations that affect model

Results

- Experiments in Progress
- Tested 6762 TPTP benchmarks in 39 categories
  - smt2 format, quantifiers over non-arithmetic sorts
  - z3 vs cvc4+fmf
    - SAT answers:
      - 418 SAT by z3
        - » 161 where cvc4+fmf cannot
      - 351 SAT by cvc4+fmf
        - » 93 where z3 cannot
    - cvc4+fmf wins more categories (11 to 6)
  - Current implementation uses naïve instantiation
    - Exhaustive instantiation using all combinations of terms
  - Interestingly, cvc4+fmf answers unsat where z3/cvc3 cannot
    - 75 benchmarks

Conclusion

- Finite model finding in CVC4
  - Uses solver for UF + cardinality constraints
  - Finds minimal models for ground constraints
  - Uses exhaustive instantiation
- Practical approach for SMT problems
  - Can answer SAT quickly in cases
  - Orthogonal to other approaches to quantifiers

Questions?