MathCheck: A Math Assistant Combining SAT with Computer Algebra Systems

Ed Zulkoski, Vijay Ganesh, Krzysztof Czarnecki

University of Waterloo

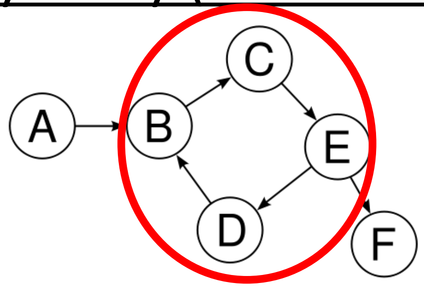
July 12, 2016

UNIVERSITY OF WATERLOO

Problem Statement

Many problems have an underlying Boolean structure, but are **not easily expressed** using standard SAT/SMT solvers.

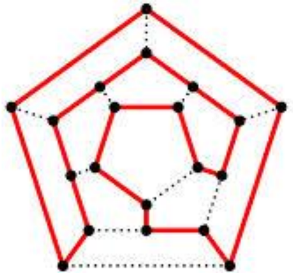
Acyclicity (Gebser'14)



Constrained Clustering (Métivier'12)

	A	B	C	D	E	F
t_1	×		×	×		
t_2		×			×	
t_3	×		×	×		
t_4		×	×	×		×
t_5		×			×	
t_6		×			×	
t_7		×		×		×

Hamiltonicity (Velev'09)

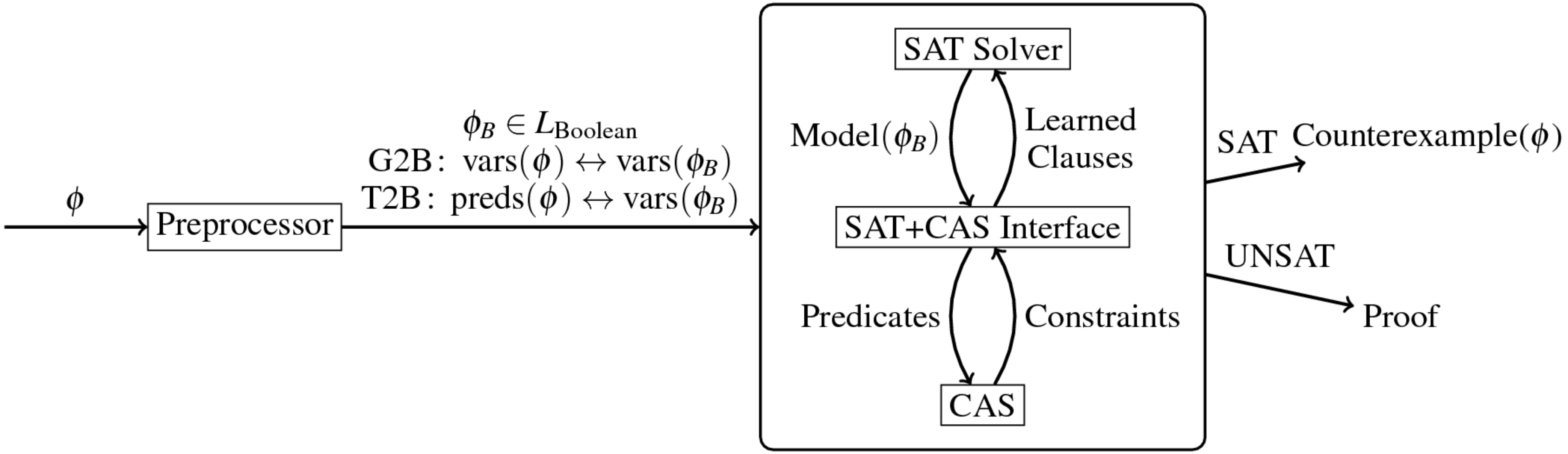


Finite domain search + complex predicates.

Goals

- Computer algebra systems (CAS) contain SOTA algorithms for solving complex properties
- SAT solvers are one of the best general approaches for finite domain search
- **Goal 1**: incorporate algorithms from a CAS with a SAT solver for:
    - Counterexample Construction for Math Conjectures
    - Bug finding
- **Goal 2**: design an easily extensible language/API for such a system
    - Current focus is on graph theory

DPLL(CAS) Architecture

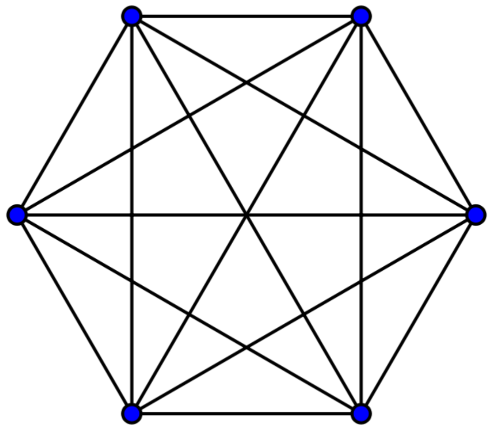


Extensibility preferred to a "one-algorithm-fits-all" approach.

Graph Variable Representation

graph x(6)

- One Boolean per each potential vertex
- One Boolean per each potential edge

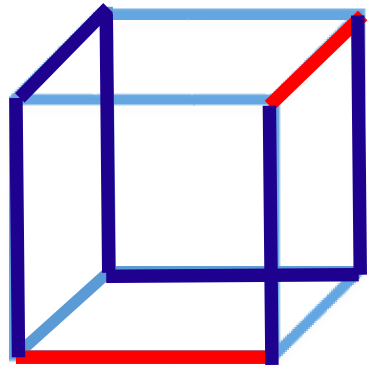


- Mapping between graph components and Booleans to facilitate defining SAT-based graph constraints

Case Study: Ruskey-Savage Conjecture

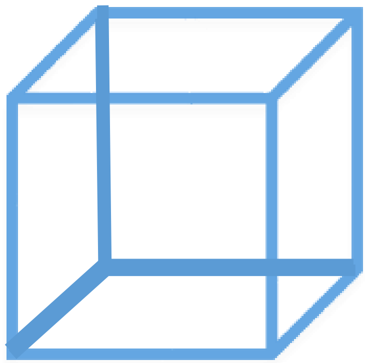
Conjecture: For every $d \geq 2$, any matching of the hypercube Q_d extends to a Hamiltonian cycle.

- Matching – independent set of edges that share no vertices
  - Maximal – cannot add edges without violating the matching property
  - Perfect – it covers all vertices
- Hamiltonian cycle – cycle that touches every vertex
- Previously shown true for $d \leq 4$



Q_3

Case Study Specification ($d = 5$)



graph x(32)

sage.CubeGraph G(5)

//$\forall x. matching(x, G) \Rightarrow extends\_to\_hamiltonian(x, G)$

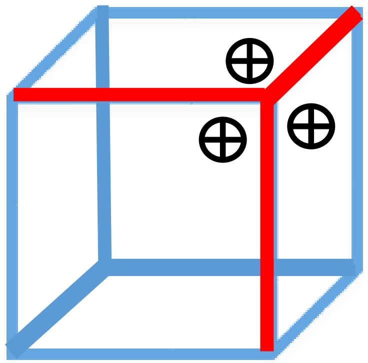
assert(matching(x,G) ∧

imperfect_matching(x,G) ∧

maximal_matching(x,G)),

query(extends_to_Hamiltonian_cycle(x,G))

Case Study Specification ($d = 5$)



graph x(32)

sage.CubeGraph G(5)

$//\forall x. matching(x, G) \Rightarrow extends\_to\_hamiltonian(x, G)$

assert(matching(x,G) ∧

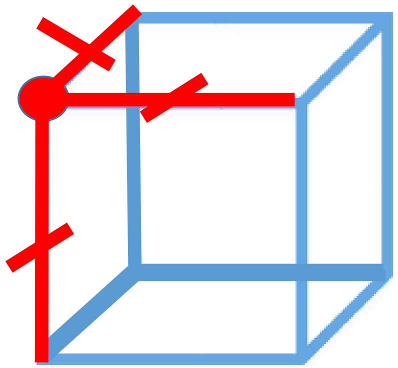
imperfect_matching(x,G) ∧

maximal_matching(x,G)),

query(extends_to_Hamiltonian_cycle(x,G))

Blasted to SAT

Case Study Specification ($d = 5$)



graph x(32)

sage.CubeGraph G(5)

//$\forall x. matching(x, G) \Rightarrow extends\_to\_hamiltonian(x, G)$

assert(matching(x,G) ∧

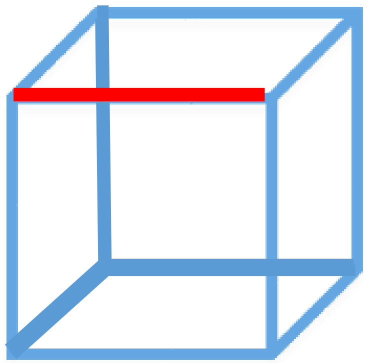
imperfect_matching(x,G) ∧

maximal_matching(x,G)),

query(extends_to_Hamiltonian_cycle(x,G))

Blasted to SAT

Case Study Specification ($d = 5$)



graph x(32)

sage.CubeGraph G(5)

//$\forall x.\, matching(x, G) \Rightarrow extends\_to\_hamiltonian(x, G)$

assert(matching(x,G) ∧

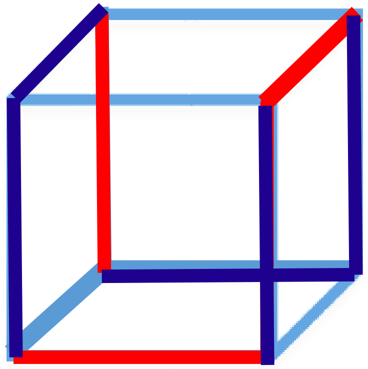
imperfect_matching(x,G) ∧

maximal_matching(x,G)),

query(extends_to_Hamiltonian_cycle(x,G))

Blasted to SAT

Case Study Specification ($d = 5$)



graph x(32)

sage.CubeGraph G(5)

//$\forall x. matching(x, G) \Rightarrow extends\_to\_hamiltonian(x, G)$

assert(matching(x,G) ∧

imperfect_matching(x,G) ∧

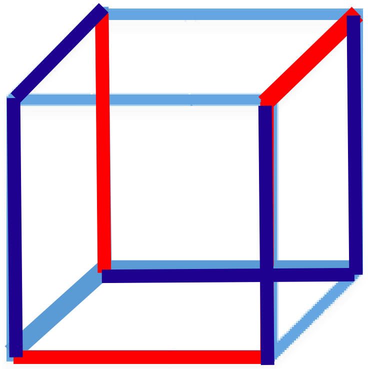
maximal_matching(x,G)),

query(extends_to_Hamiltonian_cycle(x,G))

Blasted to SAT

Checked with SAGE

Case Study Specification ($d = 5$)



graph x(32)

sage.CubeGraph G(5)

$//\forall x. matching(x, G) \Rightarrow extends\_to\_hamiltonian(x, G)$

assert(matching(x,G) ∧ ~10 LOC

imperfect_matching(x,G) ∧ ~5 LOC

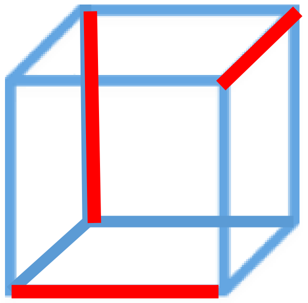
maximal_matching(x,G)), ~5 LOC

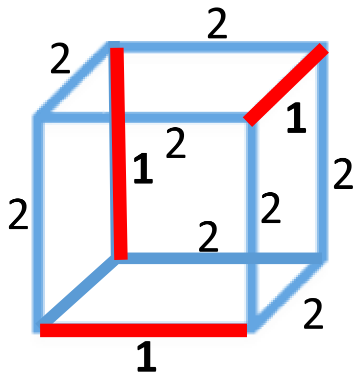
Blasted to SAT

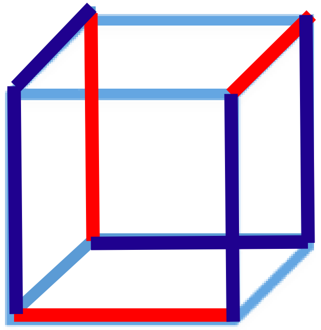
query(extends_to_Hamiltonian_cycle(x,G)) ~25 LOC

Checked with SAGE

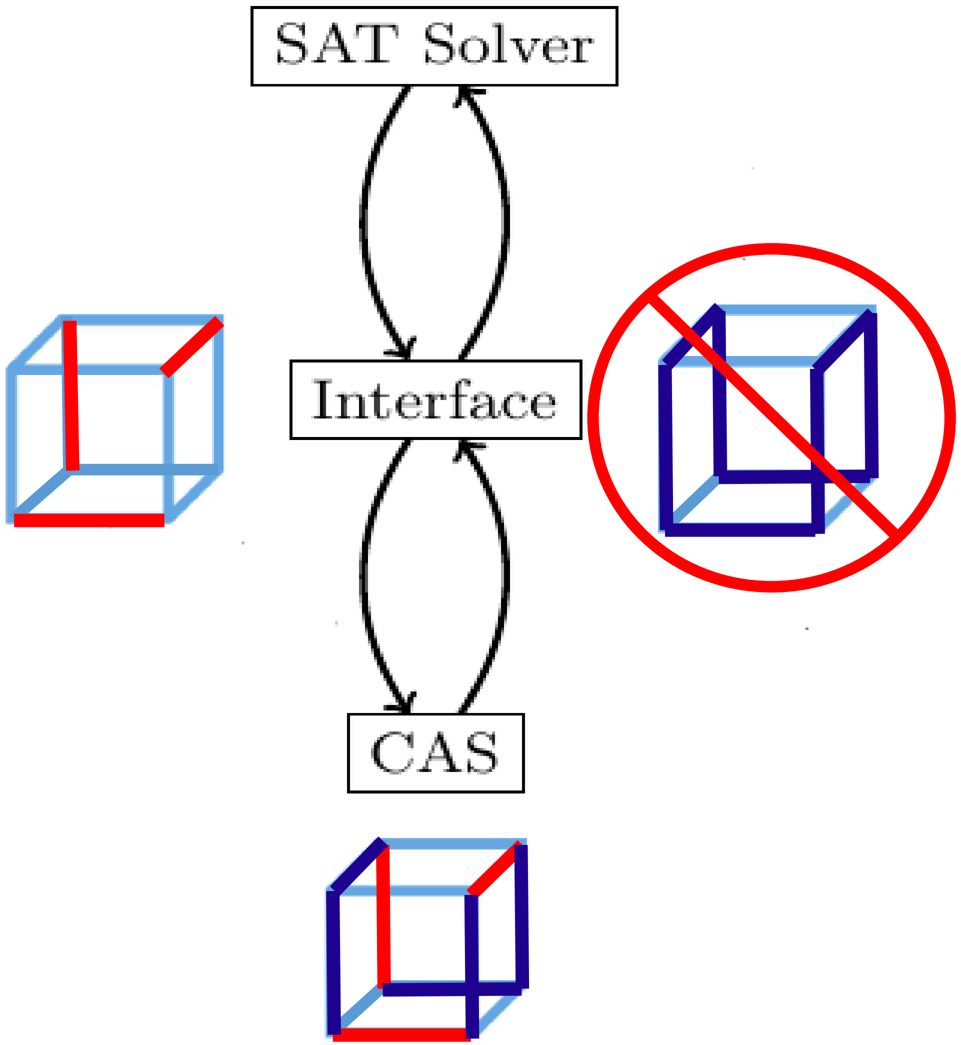
```
ExtendsToHamiltonian()
    x ← s.getGraph(G)
    q ← CubeGraph(5)
    for e in q.edges() do
        if e in g
            q.setEdgeLabel(e, 1)
        else
            q.setEdgeLabel(e, 2)
        ⟨cycle, weight⟩ ← TSP(q)
        if weight == 2 · q.order() − |x|
            return True
        else
            return False
```







Case Study Approach



- Unsat after ~8 hours on laptop (Conjecture holds for $\boldsymbol{d = 5}$)
- For a pure SAT encoding, we need encode non-trivial Hamiltonicity constraints

A Sage-only approach…

- Without SAT, we need a problem-specific search routine

	#Checks of extends_to_Hamiltonian_cycle
Matchings	13,803,794,944
Imperfect Matchings	4,619,529,024
Maximal Imperfect Matchings	6,911,604
SAT Approach	**384,000**

- A Sage-only approach is:
    - Potentially less efficient
    - Potentially more error-prone