MathCheck: A Math Assistant
Combining SAT with Computer Algebra Systems

Ed Zulkoonski, Vijay Ganesh, Krzysztof Czarnecki
University of Waterloo
July 12, 2016
Many problems have an underlying Boolean structure, but are not easily expressed using standard SAT/SMT solvers.

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 $d \leq 4$
Case Study Specification ($d = 5$)

graph x(32)
sage.CubeGraph G(5)
//∀x. matching(x, G) ⇒ 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 $G(5)$

// $\forall x. \text{matching}(x, G) \Rightarrow \text{extends_to_hamiltonian}(x, G)$

assert( $\text{matching}(x, G)$ $\land$ $\text{imperfect_matching}(x, G)$ $\land$ $\text{maximal_matching}(x, G)$ ),

query( $\text{extends_to_Hamiltonian_cycle}(x, G)$ )
Case Study Specification ($d = 5$)

graph x(32)
sage.CubeGraph G(5)

// $\forall x. \text{matching}(x, G) \Rightarrow \text{extends_to_hamiltonian}(x, G)$

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)
// ∀x. matching(x, G) ⇒ 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)

\[\forall x. \text{matching}(x, G) \Rightarrow \text{extends_to_hamiltonian}(x, G)\]

assert( matching(x,G) \land 
imperfect\_matching(x,G) \land 
maximal\_matching(x,G) ),

query( extends\_to\_Hamiltonian\_cycle(x,G) )

Blasted to SAT

Checked with SAGE
Case Study Specification ($d = 5$)

```plaintext
graph x(32)
sage.CubeGraph G(5)
//∀x. matching(x, G) ⇒ 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
1: `EXTENDS_TO_HAMILTONIAN()`
2: \[x \leftarrow s.\text{getGraph}(G)\]
3: \[q \leftarrow \text{CubeGraph}(5)\]
4: \[\text{for } e \text{ in } q.\text{edges()} \text{ do}\]
5: \[\quad \text{if } e \text{ in } g\]
6: \[\quad q.\text{setEdgeLabel}(e, 1)\]
7: \[\quad \text{else}\]
8: \[\quad q.\text{setEdgeLabel}(e, 2)\]
9: \[\langle \text{cycle, weight} \rangle \leftarrow \text{TSP}(q)\]
10: \[\quad \text{if } \text{weight} == 2 \cdot q.\text{order()} - |x|\]
11: \[\quad \text{return True}\]
12: \[\quad \text{else}\]
13: \[\quad \text{return False}\]
Case Study Approach

• Unsat after ~8 hours on laptop (Conjecture holds for $d = 5$)
• 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

<table>
<thead>
<tr>
<th></th>
<th>#Checks of extends_to_Hamiltonian_cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matchings</td>
<td>13,803,794,944</td>
</tr>
<tr>
<td>Imperfect Matchings</td>
<td>4,619,529,024</td>
</tr>
<tr>
<td>Maximal Imperfect Matchings</td>
<td>6,911,604</td>
</tr>
<tr>
<td>SAT Approach</td>
<td>384,000</td>
</tr>
</tbody>
</table>

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