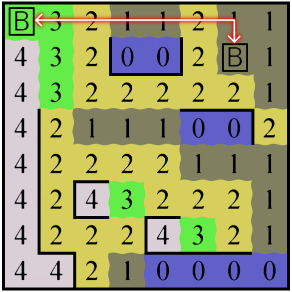
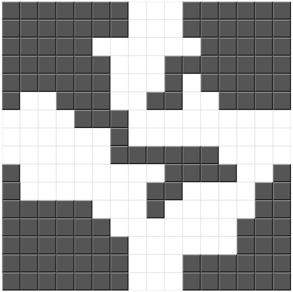
SAT Modulo Monotonic Theories

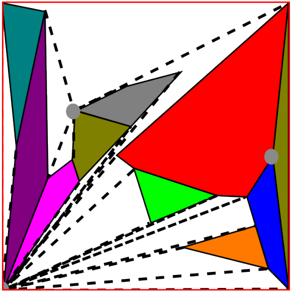
Sam Bayless*, Noah Bayless†, Holger H. Hoos*, Alan J. Hu*

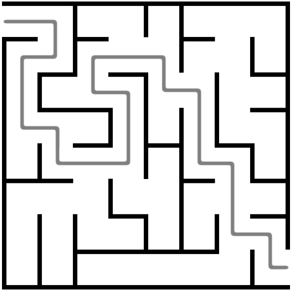
*University of British Columbia
†Point Grey Secondary School

Procedural Content Generation



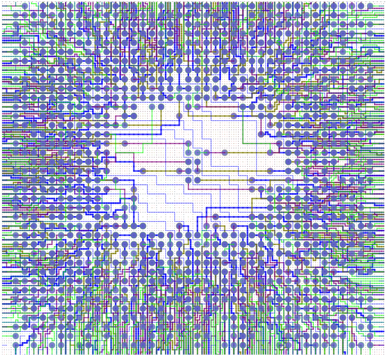




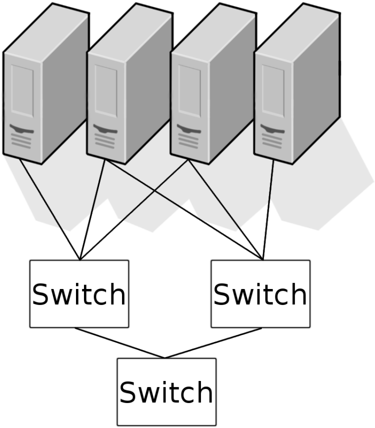


MonoSAT Applications

Circuit Layout

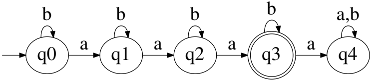


Data Center Allocation

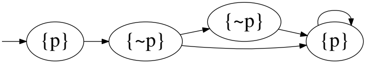


MonoSAT Applications

Finite State Machine Synthesis



CTL Controller Synthesis



MonoSAT

MonoSAT is a SAT Modulo Theory Solver for:

- Graph Predicates:
  - Reachability
  - Shortest paths
  - Maximum $s - t$ flow
  - Minimum Spanning Tree
  - Acyclicity

MonoSAT

MonoSAT is a SAT Modulo Theory Solver for:

- Graph Predicates:
  - Reachability
  - Shortest paths
  - Maximum $s - t$ flow
  - Minimum Spanning Tree
  - Acyclicity
- Collision Detection for Convex Hulls
- Finite State Machine String Acceptance
- L-Systems, Boolean Geometry, CTL checking (soon)

These are all *monotonic theories*

Boolean Monotonic Theories

A function p is a *Boolean monotonic predicate* iff:

1. : p returns a Boolean
2. : All arguments are of p are Boolean
3. : $p(\ldots, \mathrm{F}, \ldots) \implies p(\ldots, \mathrm{T}, \ldots)$

Boolean Monotonic Theories

A function p is a *Boolean monotonic predicate* iff:

1. : p returns a Boolean
2. : All arguments are of p are Boolean
3. : $p(\ldots, \mathrm{F}, \ldots) \implies p(\ldots, \mathrm{T}, \ldots)$

Definition (Boolean Monotonic Theory)

A theory T with signature Σ is Boolean monotonic if and only if:

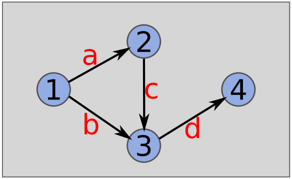
1. The only sort in Σ is Boolean;
2. all predicates in Σ are monotonic; and
3. all functions in Σ are monotonic.

Graph Constraints in SMMT

A formula over Booleans, edges, and monotonic predicates:

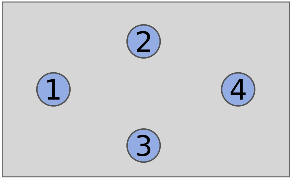
$$(a \lor \neg b) \land (b \lor c) \land (\neg c \lor \neg d) \land (reaches_{1,3} \lor reaches_{1,4})$$

And 1 or more symbolic graphs:



Graph Constraints in SMMT

'Reachability' is Boolean monotonic:

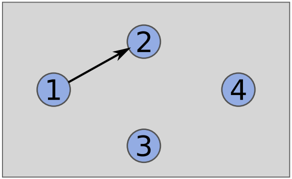


$$reaches_{1,3} \mapsto False$$

$$reaches_{1,4} \mapsto False$$

Graph Constraints in SMMT

'Reachability' is Boolean monotonic:

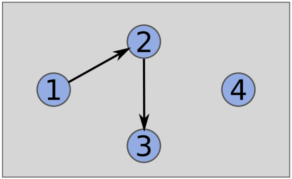


$$reaches_{1,3} \mapsto False$$

$$reaches_{1,4} \mapsto False$$

Graph Constraints in SMMT

'Reachability' is Boolean monotonic:

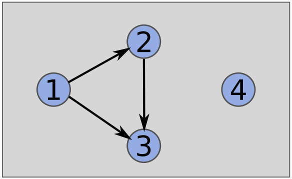


$$reaches_{1,3} \mapsto True$$

$$reaches_{1,4} \mapsto False$$

Graph Constraints in SMMT

'Reachability' is Boolean monotonic:

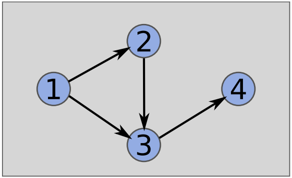


$$reaches_{1,3} \mapsto True$$

$$reaches_{1,4} \mapsto False$$

Graph Constraints in SMMT

'Reachability' is Boolean monotonic:



$$reaches_{1,3} \mapsto True$$

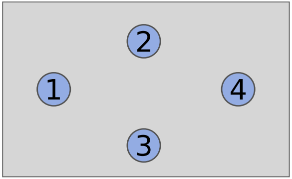
$$reaches_{1,4} \mapsto True$$

Theory Propagation in SMMT

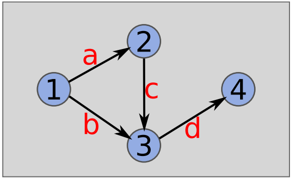
Formula:

$$(a \lor \neg b) \land (b \lor c) \land (\neg c \lor \neg d) \land (reaches_{1,3} \lor reaches_{1,4})$$

Assignment:



Underapproximation



Overapproximation

$reaches_{1,3}$

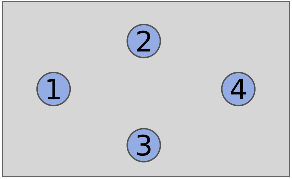
$reaches_{1,4}$

Theory Propagation in SMMT

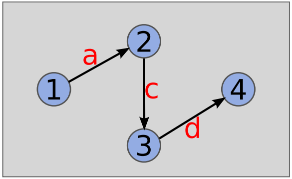
Formula:

$$(a \vee \neg b) \wedge (b \vee c) \wedge (\neg c \vee \neg d) \wedge (reaches_{1,3} \vee reaches_{1,4})$$

Assignment: $\neg b$



Underapproximation



Overapproximation

$reaches_{1,3}$

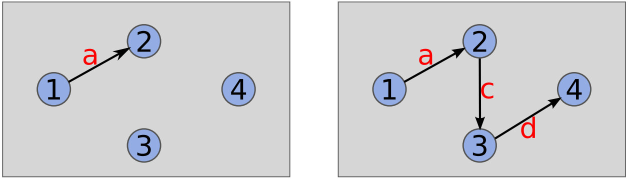
$reaches_{1,4}$

Theory Propagation in SMMT

Formula:

$$(a \vee \neg b) \wedge (b \vee c) \wedge (\neg c \vee \neg d) \wedge (reaches_{1,3} \vee reaches_{1,4})$$

Assignment: $\neg b, a$



Underapproximation   Overapproximation

$reaches_{1,3}$

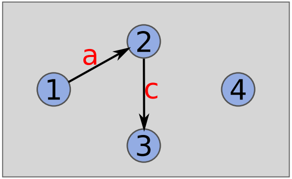
$reaches_{1,4}$

Theory Propagation in SMMT

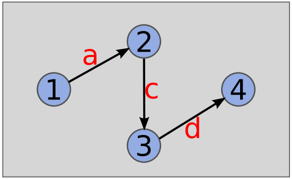
Formula:

$$(a \vee \neg b) \wedge (b \vee c) \wedge (\neg c \vee \neg d) \wedge (reaches_{1,3} \vee reaches_{1,4})$$

Assignment: $\neg b, a, c$



Underapproximation



Overapproximation

$$reaches_{1,3} \mapsto True$$

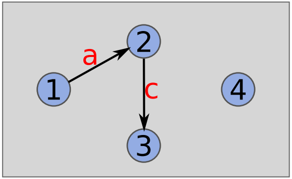
$$reaches_{1,4}$$

Theory Propagation in SMMT

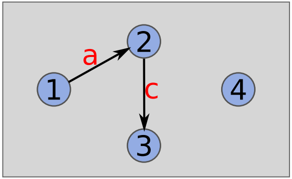
Formula:

$$(a \vee \neg b) \wedge (b \vee c) \wedge (\neg c \vee \neg d) \wedge (reaches_{1,3} \vee reaches_{1,4})$$

Assignment: $\neg b, a, c, \neg d$



Underapproximation



Overapproximation

$$reaches_{1,3} \mapsto True$$

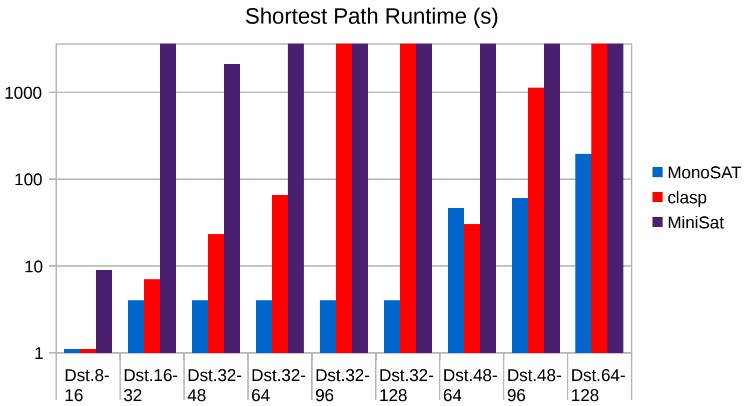
$$reaches_{1,4} \mapsto False$$

Theory Propagation in SMMT

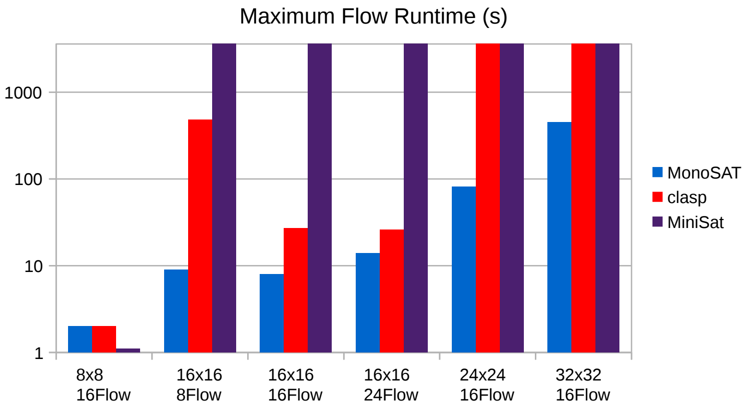
Theory propagation in SMMT has useful properties:

- Easy to implement.
- Can use off-the-shelf algorithms.
- Improved worst-case clause learning.

MonoSAT Applications: Shortest Path Constraints

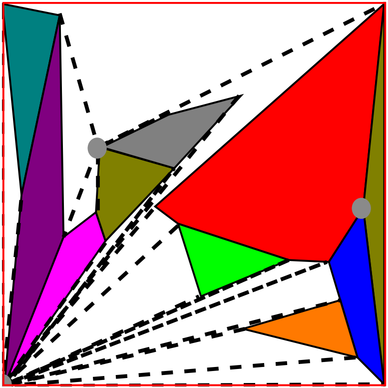


MonoSAT Applications: Maximum Flow Constraints



MonoSAT Applications: Convex Hull Containment

Art Gallery Synthesis	MonoSAT	Z3
10 points, 3 hulls, $\leq$3 cameras	**2s**	7s
20 points, 4 hulls, $\leq$4 cameras	**36s**	433s
30 points, 5 hulls, $\leq$5 cameras	**187s**	$>$ 3600s
40 points, 6 hulls, $\leq$6 cameras	**645s**	$>$ 3600s
50 points, 7 hulls, $\leq$7 cameras	**3531s**	$>$ 3600s



Conclusion

- Monotonic theories have *many* applications.
- Building SMT solvers for them is easy.
- MonoSAT supports many graph properties (and more!), and it is free & open-source:
  - *New!* Bit vector support,
  - *New!* Python support.

Website: `cs.ubc.ca/labs/isd/Projects/monosat`

GitHub: `github.com/sambayless/monosat`

MonoSAT + Python

```
from monosat import *

a = Var()
b = Var()
c = Or(a, Not(b))

Assert(c)

result = Solve()
```

MonoSAT + Python

```
from monosat import *
g= Graph()
e1 = g.addEdge(1,2)
e2 = g.addEdge(2,3)
e3 = g.addEdge(1,3)

Assert(Not(And(e1,e3)))

Assert(g.reaches(1,3))

result = Solve()
```

MonoSAT + Python

```
from monosat import *
g= Graph()
e1 = g.addEdge(1,2)
e2 = g.addEdge(2,3)
e3 = g.addEdge(1,3)

Assert(Or(g.reaches(1,3),
    g.distance_leq(1,3,2)))

result = Solve()
```

MonoSAT + Python

```
from monosat import *
g= Graph()
bv1 = BitVector(4)
bv2 = BitVector(4)
bv3 = BitVector(4)
e1 = g.addEdge(1,2,bv1)
e2 = g.addEdge(2,3,bv2)

Assert(g.distance_leq(1,3,bv3))
Assert(Not(g.distance_lt(1,3,bv3)))

Assert((bv1 + bv3) == 9)

result = Solve()
```