



Deciding Unbounded Heaps in an SMT Framework

Zvonimir Rakamarić¹, Roberto Bruttomesso², Alan Hu¹,
Alessandro Cimatti²

¹University of British Columbia; ²ITC-IRST

5th International Workshop on Satisfiability Modulo Theories
July 1, 2007

Introduction

- Heap-manipulating programs (HMPs)
 - Manipulate unbounded heap structures via pointers
- Many specialized theories for verification of HMPs
 - Great for heap properties (unbounded reachability)
 - Poor handling of data (integers, reals, linear arithmetic)
 - Real code - we need both!
- SMT solvers
 - Widely used in software verification
 - Many important theories (linear arith., arrays, ...)
 - None supports a theory for HMP verification
 - Real code - we need both!

Introduction

■ Solution

- Integrate a heap theory into an SMT solver

■ Logic for HMPs

- Expressive enough logic
- Efficient decision procedure
- Only boolean data

■ MathSAT

- Supports many required theories
- Easy integration of new theories via Delayed Theory Combination (DTC)

■ Contributions

- Integration
- Experiments – it actually works (usable and efficient)



Overview

- Introduction
- Logic for HMPs
- Delayed Theory Combination (DTC)
- Experiments
- Conclusion

Overview

- Introduction
- Logic for HMPs
- Delayed Theory Combination (DTC)
- Experiments
- Conclusion

Logic for HMPs - Syntax

$c \in Constants$

$x \in DataVariables$

$d, d' \in DataFields$

$v \in PointerVariables$

$f, f' \in PointerFields$

$NodeTerm ::= v \mid next(f, NodeTerm)$

$DataTerm ::= c \mid x \mid data(d, NodeTerm)$

$Atom ::= NodeTerm = NodeTerm \mid$
 $DataTerm = DataTerm \mid$
 $reach(f, NodeTerm, NodeTerm) \mid$
 $between(f, NodeTerm, NodeTerm, NodeTerm)$

$Literal ::= Atom \mid \sim Atom \mid$
 $update_pfield(f, NodeTerm, NodeTerm, f') \mid$
 $update_dfield(d, NodeTerm, DataTerm, d')$

$Formula ::= Lit. \mid Formula \wedge Formula \mid Formula \vee Formula$

Logic for HMPs

■ Theory of Data Fields

- { =, data, update_dfield }

- Handled by MathSAT

 - EUF + update axioms

- Currently only boolean and integer data fields

 - Easily extendable to other MathSAT's data types

■ Theory of Unbounded Reachability over Heaps

- { =, next, reach, between, update_pfield }

- Handled by the saturation based decision procedure described in previous work

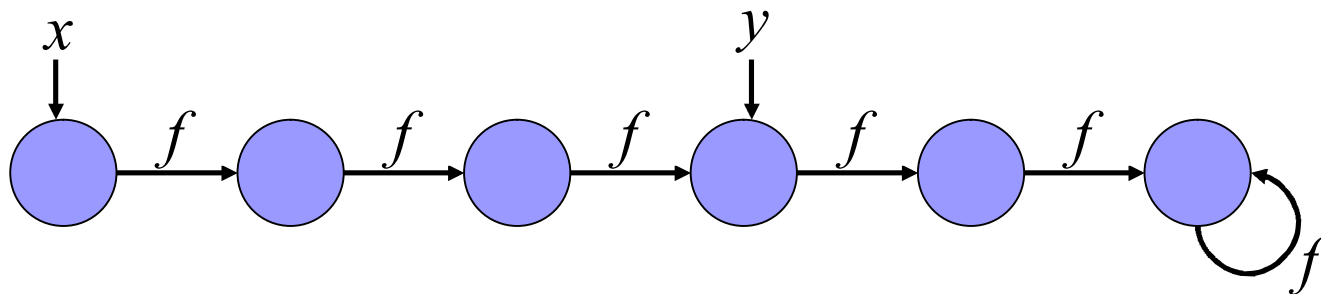
■ Signatures disjoint from other MathSAT's theories

Theory of Unbounded Reachability

- Semantics defined over *heap structures*
 - Set of heap nodes
 - Set of pointer variables
 - Pointers to heap nodes
 - Set of pointer fields
 - Links between heap nodes
- Reachability and between atoms
- Stably-infinite
- Non-convex

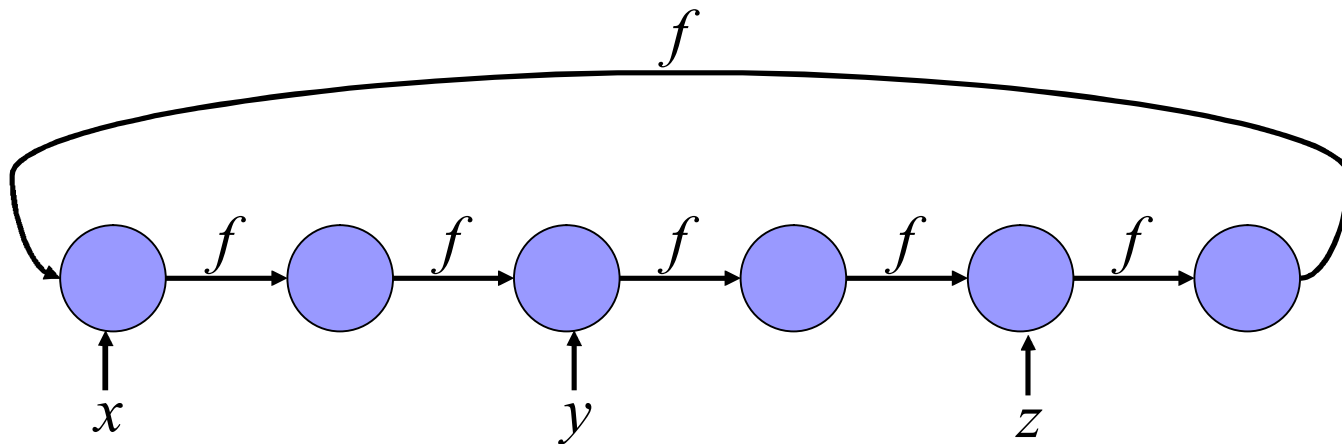
Unbounded Reachability

- $\text{reach}(f, x, y)$ – unbounded reachability (i.e. transitive closure)
 - Node y is reachable from node x following 0 or more f pointer fields



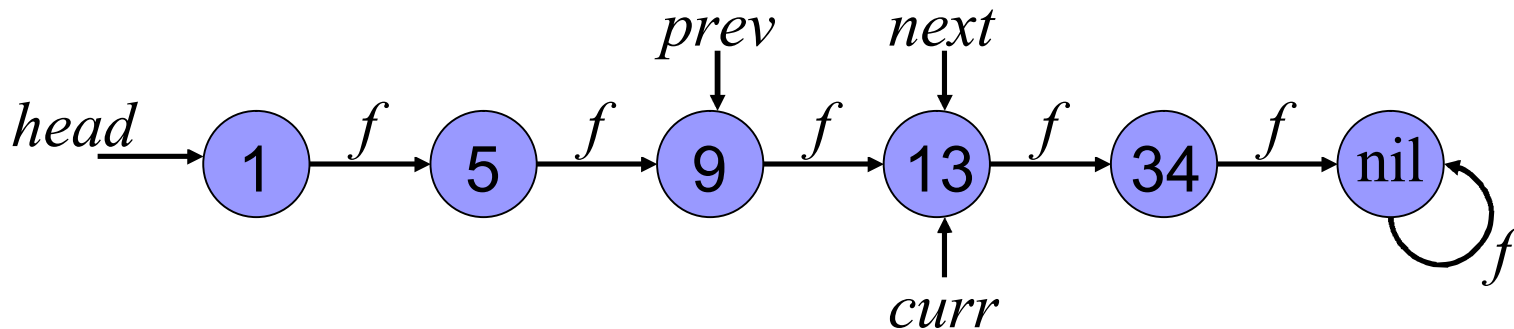
Between Atom

- $\text{between}(f, x, y, z)$
 - Node y is between nodes x and z following f pointer fields
 - Crucial for expressing necessary properties about cyclic lists



Example

- Logic for HMPs with integer data fields and linear arithmetic over integers



Some true literals:

$next = curr$

$data(d, curr) = 13$

$data(d, curr) = data(d, prev) + 4$

$reach(f, head, prev)$

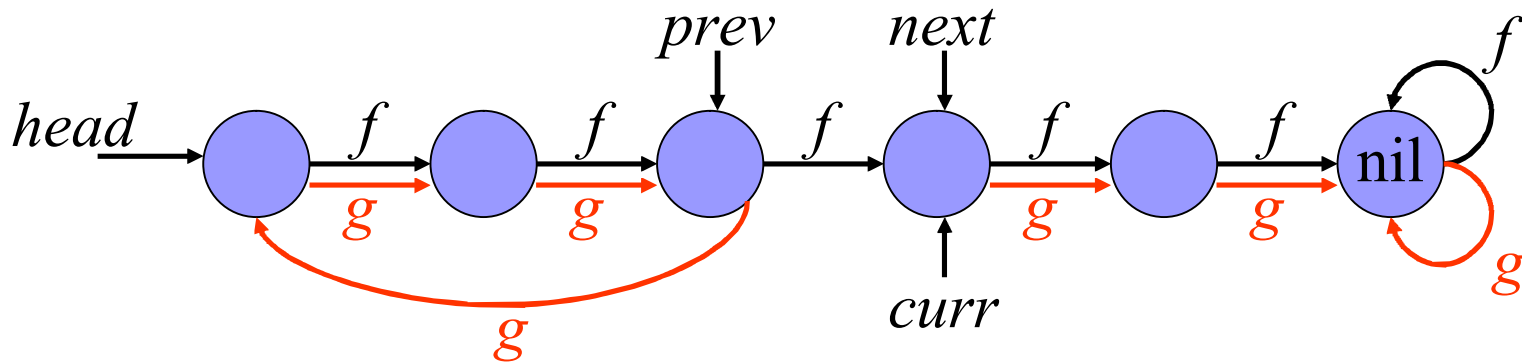
Some false literals:

$next = nil$

$data(d, prev) > data(d, curr)$

$reach(f, next, prev)$

Example with Update



True literals:

$\text{update_pfield}(f, prev, head, g)$

$\text{update_pfield}(g, prev, next, f)$

False literals:

$\text{update_pfield}(f, prev, nil, g)$

Overview

- Introduction
- Logic for HMPs
- Delayed Theory Combination (DTC)
- Experiments
- Conclusion

Delayed Theory Combination

- Uses boolean engine for communication between theory solvers
 - Eagerly introduce the set of all possible equality atoms between shared variables
 - The set contains interface equalities that theory solvers might need to exchange
 - The communication is emulated by the enumeration of all possible interface equalities

Delayed Theory Combination

■ Advantages

- Integration is implicitly handled at the boolean level and not at the solver level
 - No need to build a Nelson-Open “box” around theory solvers
- Disjunction in case of non-convexity is automatically handled at the boolean level
- Theory solvers don't need deduction capabilities

Integration

- Easily accomplished because of advantages of DTC
 - Almost no changes to the decision procedure or its interface
- Data updates $\text{update_dfield}(d, t, v, d')$ are handled by adding eagerly the following set of axioms
$$\{d'(t) \approx v\} \cup \{d'(s) \approx d(s) \mid s \in \text{NodeTerm}, s \neq t\}$$
where \approx is the equality = for integer data and \leftrightarrow for boolean data.



Overview

- Introduction
- Logic for HMPs
- Delayed Theory Combination (DTC)
- **Experiments**
- Conclusion

Experiments

- Tested extended MathSAT using queries generated from predicate abstraction of HMP examples
- Comparison with pure unbounded reachability decision procedure from previous work
- HMPs we couldn't handle before
- Three sets of results
 - No data (old DP and MathSAT)
 - Boolean data (old DP and MathSAT); Integer data, difference logic (MathSAT)
 - Integer data, linear arithmetic (MathSAT)

Old DP vs. MathSAT – No Data

Program	Old DP (s)	MathSAT (s)
LIST-REVERSE	0.2	0.2
LIST-ADD	0.1	0.1
ND-INSERT	0.5	0.6
ND-REMOVE	0.9	1.2
ZIP	17.3	27.3
CREATE-INSERT	14.8	15.6
CREATE-FREE	457.4	489.2
REMOVE-DOUBLY	24.3	33
REMOVE-CYCLIC-DOUBLY	15.6	15.7
LINUX-LIST-ADD	6.4	8.9
LINUX-LIST-ADD-TAIL	7.3	10
LINUX-LIST-DEL	24.7	25.2

Old DP vs. MathSAT

Program	Boolean		Integer
	Old DP	MathSAT	MathSAT
SORTED-ZIP	22.8	46.2	53.9
SORTED-INSERT	13.8	25.3	40.4
BUBBLE-SORT *	11.1	16.5	16.9
BUBBLE-SORT *	114.9	209	371.3
REMOVE-ELEMENTS	8.8	14.9	16.4
REMOVE-SEGMENT	2.2	10	10.3
SEARCH-AND-SET	5.3	10.8	13.7
SET-UNION *	1.4	2.2	5.8
CREATE-INSERT-DATA	39.7	47.3	53.6
INIT-LIST	0.1	0.1	0.1
INIT-LIST-VAR	0.2	0.4	0.4
INIT-CYCLIC	0.2	0.4	0.4
SORTED-INSERT-DNODES	77.9	108.1	175.7

New Examples

Program	MathSAT (s)
LAZY-SIMPLE	33.4
LAZY-SIMPLE-BACKW	2.2
INIT-INCREMENT	1.6
INIT-ADD	1.8
INIT-ADD-FLAG	1.4
INIT-MULT	1.8

Comparison

- Reasonable performance penalty
 - Could be improved with better predicate abstraction
 - Queries are conjunctions of literals - no boolean structure
 - Large number of small queries
 - No benefit from MathSAT's enumeration heuristics
- Completely new examples we couldn't handle before
 - Use combination of theories

Overview

- Introduction
- Logic for HMPs
- Delayed Theory Combination (DTC)
- Experiments
- Conclusion

Conclusion

- Integrated the unbounded reachability theory into MathSAT
 - First solver that supports such a theory
 - Access to a rich set of theories MathSAT supports
 - Easy because of DTC
- The integration works
 - Verified HMP examples we couldn't handle before
 - Reasonable performance penalty
- Available at <http://mathsat.itc.it>



The End

Thank you!

HMP Example

```
1: procedure INIT-ADD-FLAG(head, val)
2:   assume  $next^*(head, t) \wedge next^*(head, nil) \wedge \neg t = nil \wedge oldData = data(t) \wedge oldFlag =$ 
       $flag(t)$ 
3:   curr := head;
4:   while  $\neg curr = nil$  do
5:     if  $\neg flag(curr)$  then
6:        $data(curr) := data(curr) + val$ ;
7:        $flag(curr) := true$ ;
8:     end if
9:     curr := next(curr);
10:  end while
11:  assert  $next^*(head, t) \wedge next^*(head, nil) \wedge \neg t = nil \wedge flag(t) \wedge (oldFlag \vee data(t) =$ 
       $oldData + val)$ 
12: end procedure
```

MathSAT Query Example 1

```
VAR curr, t : H_NODE
```

```
VAR tmpi : INTEGER
```

```
CONST d
```

```
FORMULA data_int(t,d)=tmpi+2 &  
       data_int(curr,d)=tmpi+5 & curr=t
```

MathSAT Query Example 2

```
VAR nil, curr, t, head : H_NODE
```

```
VAR tmpd : BOOLEAN
```

```
VAR tmpi, tmpil : INTEGER
```

```
CONST f
```

```
CONST d, d1
```

```
FORMULA nil=curr & data_bool(curr,d1) &  
  tmpil=data_int(t,d) & ¬(tmpd<->data_bool(t,d1)) &  
  star(head,¬t,f) & star(head,nil,f) & ¬t=nil &  
  data_int(t,d)=tmpil+tmpi & data_bool(t,d1) &  
  ¬tmpd
```

MathSAT Query Example 3

```
VAR x, curr, t, nil : H_NODE
```

```
VAR _true : BOOLEAN
```

```
CONST f, d, dp
```

```
FORMULA UPDATED_INT(x, 10, d, dp)
```

```
FORMULA x=curr & curr=next(x, f) & star(x, t, f) &  
  star(next(x, f), x, f) & _true &  
  between(curr, t, x, f) & ~x=nil &  
  between(x, t, curr, f) & star(t, curr, f) & t=x &  
  data_int(t, dp)=10
```

Basic IRs

$$\frac{}{x=x} \text{IDENT} \qquad \frac{}{f^*(x,x)} \text{REFLEX} \qquad \frac{f(x)=y}{f^*(x,y)} \text{TRANS1}$$

$$\frac{f^*(x,y) \quad f^*(y,z)}{f^*(x,z)} \text{TRANS2} \qquad \frac{f(x)=y \quad f^*(x,z)}{x=z \quad f^*(y,z)} \text{FUNC}$$

$$\frac{f(x_1)=x_2 \quad f(x_2)=x_3 \quad \dots \quad f(x_k)=x_1 \quad f^*(x_1,y)}{y=x_1 \quad y=x_2 \quad \dots \quad y=x_k} \text{CYCLE}_k$$

$$\frac{f^*(x,y) \quad f^*(y,x) \quad f^*(x,z)}{x=y \quad f^*(z,x)} \text{SCC}$$

$$\frac{f^*(x,y) \quad f^*(x,z)}{f^*(y,z) \quad f^*(z,y)} \text{TOTAL}$$

$$\frac{f(x)=z \quad f(y)=z \quad f^*(x,y) \quad f^*(y,x)}{x=y} \text{SHARE}$$

$$\frac{d(x) \quad -d(y)}{-x=y} \text{NOTEQNODES}$$

Between IRs

$$\frac{}{\text{btwn}_f(x,x,x)} \text{BTWREFLEX} \quad \frac{f^*(x,y) \quad f^*(y,z) \quad f(z)=x}{\text{btwn}_f(x,y,z)} \text{BTW1}$$

$$\frac{\text{btwn}_f(x,y,z)}{f^*(x,y) \quad f^*(y,z)} \text{BTW2} \quad \frac{f(x)=w \quad \text{btwn}_f(x,y,z)}{\text{btwn}_f(w,y,z) \quad x=y} \text{BTW3}$$

$$\frac{\text{btwn}_f(x,y,z) \quad \text{btwn}_f(x,z,y)}{y=z} \text{BTW4} \quad \frac{f^*(x,y) \quad f^*(x,z)}{\text{btwn}_f(x,y,z) \quad \text{btwn}_f(x,z,y)} \text{BTW5}$$

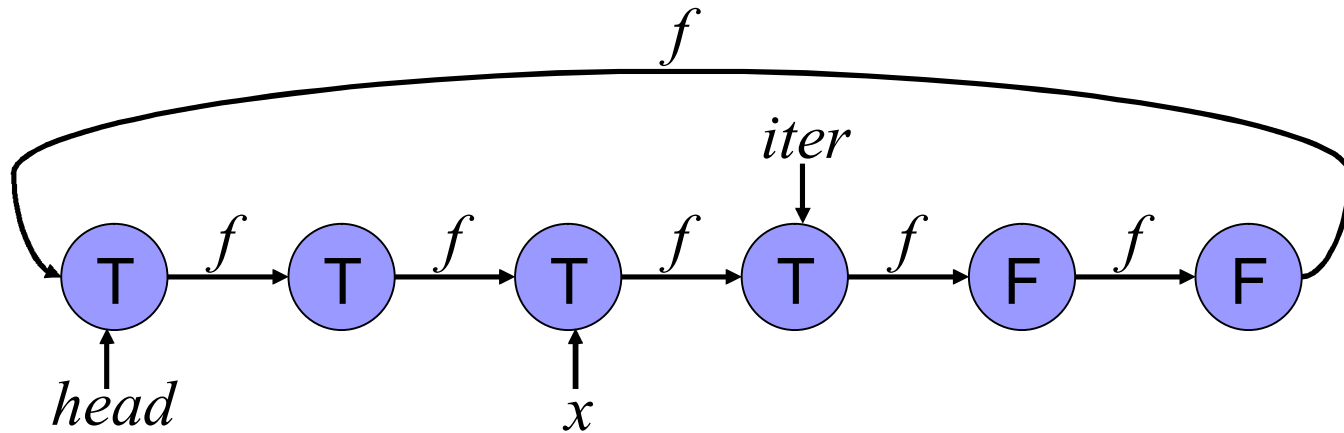
$$\frac{f^*(x,y) \quad f^*(y,z) \quad f^*(z,x)}{\text{btwn}_f(x,y,z) \quad \text{btwn}_f(x,z,y) \quad \text{btwn}_f(y,z,x) \quad \text{btwn}_f(z,y,x) \quad x=y \quad x=z \quad y=z} \text{BTW6} \quad \frac{f^*(x,y)}{\text{btwn}_f(x,x,y) \quad \text{btwn}_f(x,y,y)} \text{BTW7}$$

$$\frac{\text{btwn}_f(x,y,z) \quad f(x)=z}{y=x \quad y=z} \text{BTW8} \quad \frac{f(z)=w \quad \text{btwn}_f(x,y,w) \quad f^*(x,z)}{\text{btwn}_f(x,y,z) \quad y=w} \text{BTW9}$$

$$\frac{\text{btwn}_f(x,y,z) \quad \text{btwn}_f(w,z,y) \quad f^*(x,w)}{f^*(z,w) \quad y=z} \text{BTW10} \quad \frac{\text{btwn}_f(w,x,y) \quad \text{btwn}_f(w,y,z)}{\text{btwn}_f(w,x,z)} \text{BTW11}$$

$$\frac{\text{btwn}_f(v,u,x) \quad \text{btwn}_f(v,u,y) \quad \text{btwn}_f(u,x,y)}{\text{btwn}_f(v,x,y)} \text{BTW12} \quad \frac{\text{btwn}_f(x,y,z) \quad \neg x=z}{\text{btwn}_{f'}(x,y,z) \quad \text{btwn}_f(x,\tau_1,z) \quad \neg \tau_1=z} \text{UPDBTWN}$$

Between Example



- Typical loop invariant
 - Node x between $head$ and $iter$ has $data(d, x) = \text{true}$
- For cyclic lists the invariant can't be expressed using previous logic (i.e. using reach)
 - $\text{reach}(f, head, x) \wedge \text{reach}(f, x, iter)$ doesn't mean that x is between $head$ and $iter$
- $\text{between}(f, head, x, iter)$ solves the problem