Satisfiability of Formulas over Infinite Domains (SMT)

Mooly Sagiv
Secret Sauce – Compilation and Constraint Solving

Front End

\[ (z > 0 \land x = 1) \lor (z \leq 0 \land x = 0) \]
\[ (t > z \land y = 1) \lor (t \leq z \land y = 0) \]
\[ x \neq y \]

Constraint Solver

\[ t = 4 \]
\[ x = 6 \]
\[ y = 6 \]
\[ z = 0 \]
Boolean Satisfiability (SAT)

Is there an assignment to the $p_1, p_2, \ldots, p_n$ variables such that $\phi$ evaluates to 1?
Is there an assignment to the $x,y,z,w$ variables s.t. $\phi$ evaluates to 1?
Motivation

• We have seen that efficient SAT solvers exit
  – DPLL is the most successful complete solver

• Can we generalize the results?
  – Is “p \lor \neg q \lor (a = f(b \neg c)) \lor (g(g(b)) \neq c) \lor a\neg c \leq 7” satisfiable?

• Improve our understanding of DPLL
From Propositional to First Order Logic

• $F ::= \exists X. \ F \ | \ 
\forall X. \ F \ | \ 
F \land F \ | \ F \lor F \ | \ \neg F \ | \ r(t)$

• $t ::= f(t) \ | \ c \ | \ X$

• $r$ and $t$ are either interpreted ($<, +$) or not

• Examples:
  
  – $\forall X. \ \text{vote}(X, \ \text{trump}) \ \Rightarrow \ \exists \ Y. \ \text{vote}(Y, \ \text{clinton}) \ \land \ Y= \text{parent}(X)$
  
  – $\forall X. \ \exists \ Y. \ Y \ast Y = X$
Herbrand’s Theorem

- A first order (un-interpreted) formula F is satisfiable if and only if there exists a Herbrand model which satisfies F.

- Proof sketch:
  - Formula $\rightarrow$ Prenex Normal $\rightarrow$ Skolemize $\rightarrow$ Herbrand-Models

- High level idea
  - A first order formula is satisfiable iff a (potentially) infinite formula is satisfiable.
Satisfiability Modulo Theories

• Given a formula in first-order logic, with associated background theories, is the formula satisfiable?
  – Yes: return a satisfying solution
  – No [generate a proof of unsatisfiability]
Satisfiability Modulo Theories

• Any SAT solver can be used to decide the satisfiability of ground first-order formulas

• Often, however, one is interested in the satisfiability of certain ground formulas in a given first-order theory:
  – Pipelined microprocessors: theory of equality, atoms
    • \( f(g(a, b), c) = g(c, a) \)
  – Timed automata: planning: theory of integers/reals,
  – atoms
    • \( x - y < 2 \)
  – Software verification: combination of theories, atoms
    • \( 5 + \text{car}(a + 2) = \text{cdr}(a[j] + 1) \)

• We refer to this general problems as (ground) Satisfiability Modulo Theories, or SMT
Example Difference constraints

- Boolean combinations of `a ≤ b + k`
  - a and b are free constants
  - k ∈ Z
Uninterpreted Functions

\[
\text{read}(\text{write}(X, Y, Z), Y) = Z
\]
\[
W \neq Y \Rightarrow \text{read}(\text{write}(X, Y, Z), W) = \text{read}(X, W)
\]

\[
x + 2 = y \Rightarrow f(\text{read}(\text{write}(a, x, 3), y-2)) = f(y-x+1)
\]
A Simple Example (BMC)

Program

```c
int x;
int y=8, z=0, w=0;
if (x)
    z = y - 1;
else
    w = y + 1;
assert (z == 5 || w == 9);
```

constraints

```c
y = 8,
z = x ? y - 1 : 0,
w = x ? 0 : y + 1,
z != 5,
w != 9
```

SMT counterexample found!

```plaintext
y = 8, x = 1, w = 0, z = 7
```
Motivating Example

**Skolem-Lowenheim Formulas**

- Prenex Normal Form $\exists \forall$
- $\exists x, y \ \forall z, w : P(x, y) \land \neg P(z, w)$
Lifting SAT to SMT

- **Eager approach [UCLID]:**
  - translate into an equisatisfiable propositional formula,
  - feed it to any SAT solver

- **Lazy approach [CVC, ICS, MathSAT, Verifun, Zap]:**
  - abstract the input formula into a propositional one
  - feed it to a DPLL-based SAT solver
  - use a theory decision procedure to refine the formula

- **DPLL(T) [DPLLT, Z3, Yices, Sammy]:**
  - use the decision procedure to guide the search of a DPLL solver
(Very) Lazy Approach for SMT – Example

\[ \begin{align*}
g(a) &= c \land (f(g(a)) \neq f(c) \lor g(a) = d) \land c \neq d \\
1 & \quad \neg 2 \quad \quad 3 \quad \quad \neg 4
\end{align*} \]

Send \{1, \neg 2 \lor 3, \neg 4\} to the SAT solver

SAT solver returns \{1, \neg 2, \neg 4\}

Theory solver finds that \{1, \neg 2\} is E-unsatisfiable

Send \{1, \neg 2 \lor 3, \neg 4, \neg 1 \lor 2\} to the SAT solver

SAT solver returns \{1, 2, 3, \neg 4\}

Theory solver finds that \{1, 3, \neg 4\} is E-unsatisfiable

Send \{1, \neg 2 \lor 3, \neg 4, \neg 1 \lor 2, \neg 1 \lor \neg 3 \lor 4\} to the SAT solver

Return UNSAT
Decision Procedures

• Complete (terminating) algorithms for determining the validity (satisfiability) of a formula in a given logic
  – Cost is an issue

• Decidable logic  a logic with a decision procedure for every formula

• Decidable (computation problem)  there exists an a terminating algorithm which solves every instance of the problem
Obtaining a decision procedure

• Limit the logic
  – Only unary predicates
  – At most two variables

• Limit the class of intended models
  – Interpreted first order formulas over real

• Answer validity (satisfiability) w.r.t. a given theory
  $T \models F$
Proving Decidability

• Small model theorem
  – Every satisfiable formula has a model whose size is proportional to the size of the formula

• Direct decision procedure

• Reduction to another decidable logic
Quantifier Free First Order Logic

- Universal formulas only
- Allow a fixed scheme of first order formulas $T$
- Determine if $T \models F$
- Decidable for interesting theories
  - Uninterpreted functions
    - $\forall a, b: f(a, b) = a \Rightarrow f(f(a, b), b) = a$
  - Theory of lists
  - Arrays
- Different theories can be combined
Theory of Uninterpreted Functions  (EUF)

- Theory $T \forall X, Y: X = Y \Rightarrow f(X) = f(Y)$
- Determine the validity of universal formulas
- Decidability Ackerman 1954
- Downey, Sethi, Tarjan, Kozen, Nelson & Oppen
  Efficient Algorithms
- Bryant, German, Velev Improvements for positive terms
Small model property of EUF formulas

• Ackerman 1954

• Every satisfiable formula has a model of size $k$ where $k$ is the number of distinct function application terms

• Example
  - $x = y \lor f(g(x)) = f(g(y))$
  - $\{x, y, g(x), g(y), f(g(x)), f(g(y))\}$

• Impractical algorithm
Proof by Refutation

• Determine the validity of a formula by checking the satifiability of its negation

• For quantifier free it is enough to consider Conjunction of literals

• Example “∀A, B: f(A, B) = A⇒f(f(A, B), B) =A”
  – Proof that “f(a, b) = a ∧ ¬ f((f(a, b), b) = a” is not satisfiable
An efficient EUF algorithm (intuition)

• Goal prove satisfiability of
  \(t_1 = u_1 \land \ldots \land t_p = u_p \land r_1 \neq s_1 \land \ldots \land r_q \neq s_q\)

• Represent terms using DAGs

• Unify equal terms and their consequences

• Report UNSAT when contradicts inequalities

• Otherwise report SAT
The Congruent Closure Problem

• Given
  – A finite labeled directed graph G
    • Nodes are labeled by function symbols
    • Edges are labeled
  – A binary relation R on the nodes

• Two nodes are congruent under R if
  – They have the same label
  – Their arguments (outgoing neighbors) are in R (respectively)

• R is closed under congruences if all congruent nodes according to R are in R

• Compute the a minimal extension of R which is an equivalence relation and closed under congruences
Example 1

\[ f(a, b) = a \land f((f(a, b), b) \neq a \]
Example 2

\[ f(f(f(A))) = A \land f(f(f(f(f(A))))) = A \Rightarrow f(A) = A \]

\[ f(f(f(a))) = a \land f(f(f(f(f(a))))) = a \land f(a) \neq a \]
Computing Congruence Closure

• Let R be a relation which is congruence closed

• Compute the congruence closure of \( R \cup \{(u, v)\} \) by \( \text{MERGE}(u, v) \)

MERGE(u, v)

1. If \( \text{FIND}(u) = \text{FIND}(v) \) then return

2. Let \( P_u \) be the predecessors of vertices equivalent to \( u \) and \( P_v \) be the predecessors of vertices equivalent to \( v \)

3. UNION(u, v)

4. For each pair \((x, y)\) such that \( x \in P_u, y \in P_v, \text{CONGRUENT}(x, y) \) and \( \text{FIND}(x) \neq \text{FIND}(y) \) do \( \text{MERGE}(x, y) \)

CONGRUENT(u, v) = label(u) = label(v) \( \land \forall i: \text{FIND}(u[i]) = \text{FIND}(v[i]) \)
Properties of the Congruence Closure Algorithm

- Partial Correctness
- Complexity $O(m^2)$
- Downey, Sethi, and Tarjan achieves $O(m \log n)$ by storing the vertices in a hash table keyed by the list of equivalence classes of their successors.
Application 1: EUF

- construct a graph $G$ which corresponds to the set of all terms appearing in the conjunction
  $t_1 = u_1 \land \ldots \land t_p = u_p \land r_1 \neq s_1 \land \ldots \land r_q \neq s_q$
- For each term $i$ appearing in the conjunction let $\tau(i)$ denote the node of the term
- Let $R$ be the identity relation on vertices
- For every $1 \leq i \leq p$, $\text{MERGE}(\tau(t_i), \tau(u_i))$
- If for some $1 \leq j \leq q$, $\tau(r_j)$ is equivalent to $\tau(s_j)$ report UNSAT
- Otherwise report SAT
Improvements and Extensions

• Lahiri, Bryant, Goel, Talupur TACAS 2004

• Explicit Representation
  \[\text{ITE}(e_1, e_2, e_3) = (e_1 \land e_2) \lor (\neg e_1 \land e_2)\]
  \[P(T_1, T_2, \ldots, T_k)\]

• Treat `positive’ terms differently
Simple Theory of Lisp Lists

• car, cdr, cons without nil values

• Theory (axioms):

\[
\begin{align*}
\text{car}(\text{cons}(X, Y)) &= X \\
\text{cdr}(\text{cons}(X, Y)) &= Y \\
\neg \text{atom}(X) &\Rightarrow \text{cons}(\text{car}(X), \text{cdr}(X)) = X \\
\neg \text{atom}(\text{cons}(X,Y))
\end{align*}
\]

• Goal:

\[
\begin{align*}
\text{car}(X) &= \text{car}(Y) \land \text{cdr}(X) = \text{cdr}(Y) \\
&\land \neg \text{atom}(X) \land \neg \text{atom}(Y) \Rightarrow f(X) = f(Y)
\end{align*}
\]

• Use congruence closure with special equalities
Application 2: Lisp

• $v_1 = w_1 \land \ldots \land b_r = w_r \land x_1 \neq y_1 \land \ldots \land x_s \neq y_s \land \text{atom}(u_1) \land \ldots \land \text{atom}(u_q)$

• Construct a graph $G$ which corresponds to the set of all terms appearing in the conjunction

• For each term $i$ appearing in the conjunction let $\tau(i)$ denote the node of the term

• Let $R$ be the identity relation on vertices

• For every $1 \leq i \leq r$, MERGE($\tau(v_i)$, $\tau(w_i)$)

• For every vertex $u$ labeled by cons add a vertex $v$ labeled by car and a vertex $w$ labeled by cdr with out degree one s.t. $v[1] = w[1] = u$ and MERGE($v$, $u[1]$) and MERGE($v$, $u[2]$)

• If for some $1 \leq j \leq s$, $\tau(x_j)$ is equivalent to $\tau(y_j)$ report UNSAT

• If for some $1 \leq j \leq q$, $\tau(u_j)$ is equivalent to a cons node report UNSAT

• Otherwise report SAT
Integrating Values

- \(\text{car}(\text{cons}(X,Y)) = X\)
- \(\text{cdr}(\text{cons}(X,Y)) = Y\)
- \(X \neq \text{nil} \Rightarrow \text{cons}(\text{car}(X), \text{cdr}(X)) = X\)
- \(\text{cons}(X,Y) \neq \text{nil}\)
- \(\text{car}(\text{nil}) = \text{cdr}(\text{nil}) = \text{nil}\)

- Becomes NP-Hard
Theory of Arrays (Stores)

• read(write(v, i, e), j) =
  if i=j then e else read(v, j)

• write(v, i, read(v, i)) = v

• write(write(v, i, e), i, f) = write(v, i, f)

• i ≠ j ⇒ write (write (v, i, e), j, f) =
  write (write (v, j, f), I, e)

• Eliminate write and use EUF
Combining Decision Procedures

- Programming languages combine different features
  - Arithmetic
  - Data types
  - Arrays
  - ...

- Is there a way to compose decision procedures of different theories?

- Given two decidable logics is there a way to combine the logics into a decidable logic?
Bibliography

• Nelson & Oppen
  Fast Decision Procedures Based on Congruence Closure
  JACM 1979

• Stump, Dill, Barrett, Levitt
  A Decision Procedure for an Extensional Theory of Arrays
  LICS’01
Combining Decision Procedures

• Programming languages combine different features
  – Arithmetic
  – Data types
  – Arrays
  – …

• Is there a way to compose decision procedures of different theories?

• Given two decidable logics is there a way to combine the logics into a decidable logic?
Cooperating Decision Procedures
Nelson & Oppen

• Quantifier free
• Proof be refutation
• Separate the conjunct into separate conjuncts $A \land B$
such that
  – A and B use different theories
  – Only constants are shared
• If either A or B is UNSAT report UNSAT
• When A and B are SAT propagate equalities between A
  and B and repeat
Example Theories

\begin{align*}
\text{car(cons}(X, Y)) &= X \\
\text{cdr(cons}(X, Y)) &= Y \\
\neg \text{atom}(X) &\Rightarrow \text{cons(car}(X), \text{cdr}(X)) = X \\
\neg \text{atom}(\text{cons}(X, Y)) &
\end{align*}

\begin{align*}
\text{EUF} \\
X = Y &\Rightarrow f(X) = f(Y)
\end{align*}

\begin{align*}
\text{X} + 0 &= 0 \\
\text{X} + -X &= 0 & 0 \neq 1 \\
(X + Y) + Z &= X + (Y + Z) & 0 \leq 1 \\
X + Y &= Y + X \\
X \leq X \\
X \leq Y \lor Y \leq X \\
X \leq Y \land Y \leq X &\Rightarrow X = Y \\
X \leq Y \land Y \leq Z &\Rightarrow X \leq Z \\
X \leq Y &\Rightarrow X + Z \leq Y + Z
\end{align*}
A Simple Example

\[
x \leq y \land y \leq x + \text{car}(\text{cons}(0, x)) \land P(h(x) - h(y)) \land \neg P(0)
\]

\[
x \leq y
\]

\[
y \leq x + g_1
\]

\[
g_2 = g_3 - g_4
\]

\[
g_5 = 0
\]

\[
P(g_2) = \text{true}
\]

\[
P(g_5) = \text{false}
\]

\[
g_3 = h(x)
\]

\[
g_4 = h(y)
\]

\[
g_1 = \text{car}(\text{cons}(g_5, x))
\]
Equality Propagation Procedure

1. Assign conjunctions to $F_L$ and $F_F$ s.t.,
   - $F_F$ contains only $F$-literals
   - $F_L$ contains only $L$-literals
   - $F_L \land F_F$ is satisfiable iff $F$ is satisfiable

2. If either $F_L$ or $F_F$ is UNSAT report UNSAT

3. If either $F_L$ or $F_F$ entails equality not entailed by other
   add this equality and go to step 2

4. If either $F_L$ or $F_F$ entails $u_1=v_2 \lor u_2=v_2 \lor \ldots u_k=v_k$ without
   entailing any equality alone then apply the procedure
   recursively to the $k$-formulas
   $F_L \land F_F \land v_i = u_i$
   If any of these formulas is SAT return SAT

5. Return UNSAT
Notes

• Only equalities are propagated
• Requires that the theories can find all consequent equalities
• Completeness is non-obvious
• The original paper also performs simplification
Convexity

• A formula $F$ is non-convex if $F$ entails $u_1=v_2 \lor u_2=v_2 \lor \ldots \lor u_k=v_k$ without entailing any equality alone.
  – Otherwise it is convex.

• A theory is convex.

• Convex theories:
  – EUF
  – Relational linear algebra

• Non-convex theories:
  – Theory of arrays
  – Theory of reals under multiplications: $xy=0 \land z=0 \models_R x=z \lor y=z$
  – Theory of integers under $+$ and $\leq$
Hints about Completeness

- The residues of formula
  - The strongest Boolean combinations of equalities between constants entailed by the formula

<table>
<thead>
<tr>
<th>x=f(a) ∧ y=f(b)</th>
<th>a=b → x=y</th>
</tr>
</thead>
<tbody>
<tr>
<td>x+y-a-b&gt;0</td>
<td>(x=a ∧ y=b) ∧ (x=b ∧ y=a)</td>
</tr>
<tr>
<td>x=\text{write}(v, u, e)[j]</td>
<td>i=j → x=e</td>
</tr>
<tr>
<td>x=\text{write}(v, u, e)[j] ∧ y=v[j]</td>
<td>if i=j then x=e else x=y</td>
</tr>
</tbody>
</table>

Lemma 4: If A and B are formulas whose only common parameters are constant symbols then 
\( \text{RES}(A \land B) = \text{RES}(A) \land \text{RES}(B) \)
More correct account of completeness

• A theory T is stably infinite if every quantifier-free formula is T-satisable if and only if it is satisfied by a T-model A whose domain A is infinite

• For lemma 4 we require
  – The theories are disjoint
  – Both theories are stably infinite
  – Read more in Manna 2003
The residues in the simple example

\[ x \leq y \land y \leq x + \text{car}(\text{cons}(0, x)) \land P(h(x) - h(y)) \land \neg P(0) \]

<table>
<thead>
<tr>
<th>( x \leq y )</th>
<th>( P(g_2) = \text{true} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y \leq x + g_1 )</td>
<td>( P(g_5) = \text{false} )</td>
</tr>
<tr>
<td>( g_2 = g_3 - g_4 )</td>
<td>( g_3 = h(x) )</td>
</tr>
<tr>
<td>( g_5 = 0 )</td>
<td>( g_4 = h(y) )</td>
</tr>
<tr>
<td>( g_1 = g_5 \rightarrow x = y \land g_5 = g_2 \leftrightarrow g_3 = g_4 )</td>
<td>( g_2 \neq g_5 \land x = y \rightarrow g_3 = g_4 )</td>
</tr>
</tbody>
</table>

\( g_1 = \text{car}(\text{cons}(g_5, x)) \)

\( g_1 = g_5 \)
Handling Quantifiers

- The problem becomes undecidable
- Refutationally resolution based complete procedures exist and implemented (e.g., SPASS, Vampire)
  - Not guaranteed to terminate
  - Do not handle theories
- Z3 employs incomplete heuristics
  - Instantiate universal quantifiers with relevant terms
  - Can be tuned by the user
Conclusion

- Handling specialized theories yields significant improvements
  - Efficiency
  - Termination
  - Predictability

- Combination procedures are useful

- But resolution based theorem provers can still be superior in several cases
Partial Bibliography

- Nelson & Oppen
  *Simplification by Cooperating Decision Procedures*
  TOPLAS 1980

- Tinelli & Zarba
  *Combining non-stably infinite theories*
  *Journal of Automated Reasoning*, 2006

- **Simplify: A Theorem Prover for Program Checking**
  David Detlefs, Greg Nelson, James B. Saxe
  JACM 2005

- **Combining Decision Procedures (2003)**
  Zohar Manna, Calogero G. Zarba

- **Verifun: An Explicating Theorem Prover for Quantified Formulas**
  Cormac Flanagan, Rajeev Joshi, James B. Saxe
  HPL-2004-199

- Leonardo Mendonça de Moura, Nikolaj Bjørner:

- Leonardo Mendonça de Moura, Nikolaj Bjørner:
  Engineering DPLL(T) + Saturation. IJCAR 2008: 475-490