Do SAT Solvers Make Good Configurators?

Mikoláš Janota

Lero
University College Dublin
Ireland

ASPL at SPLC 2008

lero

SFI grant no. 03/CE2/I303_1
Configuration

- A feature models represent the set of products we are interested in and the dependencies between them.
- The customer selects a product in the configuration process.
- The product should fulfill the desires of the customer but must respect the constraints imposed by the feature model.
Interactive feedback

- Car
- Gear
- PowerLocks
- Body
- Gas
- Automatic
- Electric
- Engine
- Keyless

Explanation (Gear):

!Car | Gear;
Car;

Unsatisfied constraints:

Manual | Automatic | !Gear;
Electric | !Engine | Gas;
Interactive feedback

Functionality

- Feedback takes place as the choices are being made
Interactive feedback

- feedback takes place as the choices are being made
- disabling choices that do not lead to a solution

Functionality

Mikoláš Janota
Do SAT Solvers Make Good Configurators?
Interactive feedback

- feedback takes place as the choices are being made
- disabling choices that do not lead to a solution
- never let the user violate the constrains (backtrack freeness)

Mikoláš Janota  Do SAT Solvers Make Good Configurators?
Interactive feedback

- feedback takes place as the choices are being made
- disabling choices that do not lead to a solution
- never let the user violate the constrains (*backtrack freeness*)
- explaining why a value is locked
How Do We Go About This?

Use a SAT Solver

- determines the satisfiability of a given Boolean formula
- operates on Conjunctive Normal Form (CNF)
- a certification of the response is produced
- nowadays SAT solvers are very efficient
Use a SAT Solver

- determines the satisfiability of a given Boolean formula
- operates on Conjunctive Normal Form (CNF)
- a certification of the response is produced
- nowadays SAT solvers are very efficient

Assumptions

- constraints encoded in a CNF
- decisions so far encoded as a conjunction of literals

\[ \phi \equiv f_1 \land \neg f_8 \land \ldots \]
Testing all free features after each user’s decision

\textbf{Test-Vars()}

1. \textbf{foreach} \(x\) that was not assigned to by the user
2. \textbf{do} \(\text{CanBeTrue} \leftarrow \text{TEST-SAT}(\phi, x)\)
3. \(\text{CanBeFalse} \leftarrow \text{TEST-SAT}(\phi, \neg x)\)
4. \textbf{if} \(\neg \text{CanBeTrue} \land \neg \text{CanBeFalse}\)
5. \textbf{then} \textbf{error} “Unsatisfiable constraint!”
6. \textbf{if} \(\neg \text{CanBeTrue}\) \textbf{then} \text{\textbf{SET}}(x, \text{FALSE})
7. \textbf{if} \(\neg \text{CanBeFalse}\) \textbf{then} \text{\textbf{SET}}(x, \text{TRUE})
8. \textbf{if} \(\text{CanBeTrue} \land \text{CanBeFalse}\)
9. \textbf{then} \text{\textbf{RESET}}(x)
10. \textbf{else} \text{\textbf{LOCK}}(x)
Can We Improve This?

SAT

- For satisfiable queries, the SAT solver returns with a satisfying assignment.

- All the values in this assignment are satisfiable and don’t need to be queried for.
Can We Improve This?

**SAT**

- For satisfiable queries, the SAT solver returns with a satisfying assignment.
- All the values in this assignment are satisfiable and don’t need to be queried for.

**UNSAT**

- Can a negative response of the solver help in the future?
- Example

\[
\begin{align*}
  f_1 &\implies f_2 \\
  \neg f_2 &\implies \neg f_1 \\
  \vdots
\end{align*}
\]

- Recording disabled values may help with further queries.
Satisfiability with Caching

- `KnownValues` represent values known to be SAT
- `DisabledValues` represent values known to be UNSAT

Test-SAT\((\phi: \text{Formula}, \ l: \text{Literal}) : \text{Boolean}\):

1. if \( l \in \text{KnownValues} \) then return \( \text{TRUE} \)
2. if \( l \in \text{DisabledValues} \) then return \( \text{FALSE} \)
3. \( L \leftarrow \text{SAT}(\phi \land l \land \bigwedge_{k \in \text{DisabledValues}} \neg k) \)
4. if \( L \neq \text{null} \)
5. then \( \text{KnownValues} \leftarrow \text{KnownValues} \cup L \)
6. else \( \text{DisabledValues} \leftarrow \text{DisabledValues} \cup \{l\} \)
7. return \( L \neq \text{null} \)
Explanations

- The solver produces a unsatisfiable subset of given formulas.
- This may not be minimal, several techniques how to minimize.
- In the tool an iterative technique by Zhang and Malik.
Comparing to Binary Decision Diagrams (BDDs)

- It is expected to be slower than \textit{but} much less likely to choke.
- The form of the formula is preserved and hence can be used in the explanations.
- Requires CNF \textit{however} any formula can be \textit{clausified} in polynomial size.
Discussion and Future Work

Comparing to Binary Decision Diagrams (BDDs)

- It is expected to be slower than but much less likely to choke.
- The form of the formula is preserved and hence can be used in the explanations.
- Requires CNF however any formula can be clausified in polynomial size.

Future work

- Use proofs for more efficient cache discarding.
- Could this approach work for non-Boolean domains?
Discussion and Future Work

Comparing to Binary Decision Diagrams (BDDs)

- It is expected to be slower than but much less likely to choke.
- The form of the formula is preserved and hence can be used in the explanations.
- Requires CNF however any formula can be clausified in polynomial size.

Future work

- Use proofs for more efficient cache discarding.
- Could this approach work for non-Boolean domains?

A java implementation, including a SAT solver, available at kind.ucd.ie
When to Discard Caches?

- With a new decision asserted, *KnownValues* may change and thus get discarded whereas *DisabledValues* remain the same.
- When a decision is retracted, *KnownValues* remain whereas *DisabledValues* are discarded.