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
Introduction

Configuration

- a feature models represent the set of products we are interested in and the dependencies between them
- 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 takes place as the choices are being made, disabling choices that do not lead to a solution, never letting the user violate the constraints (backtrack freeness). Explaining why a value is locked.

Mikoláš Janota
Do SAT Solvers Make Good Configurators?
Interactive feedback

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
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*)
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 constraints (backtrack freeness)
- explaining why a value is locked

Functionality
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
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

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

\[ \phi \equiv f_1 \land \neg f_8 \land \ldots \]
SAT Solver for Configuration

Testing all free features after each user’s decision

\begin{algorithm}
\textbf{Test-Vars()}
\begin{algorithmic}[1]
\State \textbf{foreach} $x$ that was not assigned to by the user \Do
\State $\text{CanBeTrue} \leftarrow \text{TEST-SAT}(\phi, x)$
\State $\text{CanBeFalse} \leftarrow \text{TEST-SAT}(\phi, \neg x)$
\EndDo
\If{$\neg \text{CanBeTrue} \land \neg \text{CanBeFalse}$}
\State \textbf{error} “Unsatisfiable constraint!”
\EndIf
\If{$\neg \text{CanBeTrue}$}
\State $\text{Set}(x, \text{FALSE})$
\EndIf
\If{$\neg \text{CanBeFalse}$}
\State $\text{Set}(x, \text{TRUE})$
\EndIf
\If{$\text{CanBeTrue} \land \text{CanBeFalse}$}
\State $\text{Reset}(x)$
\State $\text{Unlock}(x)$
\Else
\State $\text{Lock}(x)$
\EndIf
\end{algorithmic}
\end{algorithm}
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.
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

\[
\begin{align*}
  f_1 &\Rightarrow f_2 \\
  \neg f_2 &
  \\
  \ldots &
  \\
  \neg f_1 &
  \\
\end{align*}
\]

- 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**$(\phi$: Formula, $l$: Literal) : Boolean

1. if $l \in \text{KnownValues}$ then return True
2. if $l \in \text{DisabledValues}$ then return 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 KnownValues $\leftarrow$ KnownValues $\cup$ $L$
6. else DisabledValues $\leftarrow$ 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 clausified in polynomial size.
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?
Comparison 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.