flea

bit(e)s and pieces

Alexander Maringele

June 15th, 2016

Hofstadter's Law: It always takes longer than you expect, even when you take into account Hofstadter's Law.

— Douglas Hofstadter, Gödel, Escher, Bach: An Eternal Golden Braid

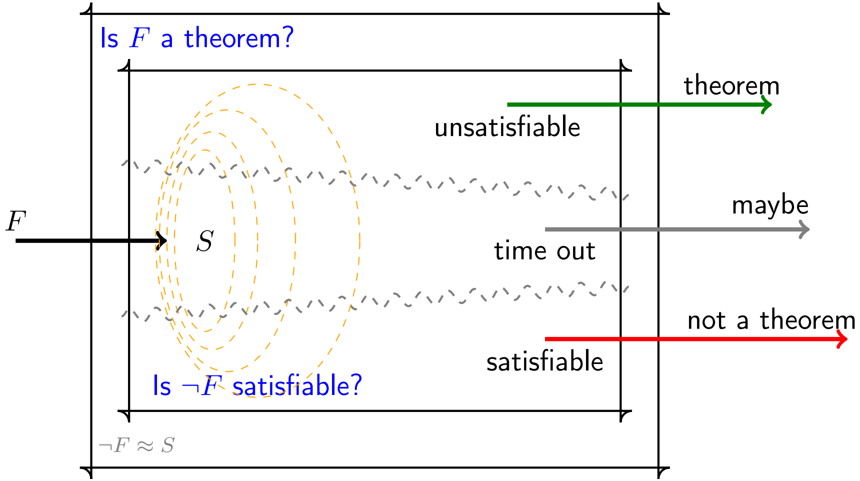
1. Previously
2. Implementation
3. Equality
4. work to do

References

Clark Barrett, Pascal Fontaine, and Cesare Tinelli, *The Satisfiability Modulo Theories Library (SMT-LIB)*, `www.SMT-LIB.org`, 2016.

Bruno Dutertre, *Yices 2.2*, Computer-Aided Verification (CAV'2014) (Armin Biere and Roderick Bloem, eds.), Lecture Notes in Computer Science, vol. 8559, Springer, July 2014, pp. 737–744.

Goal



Is F a theorem?
theorem
unsatisfiable
F
S
maybe
time out
not a theorem
satisfiable
Is $\neg F$ satisfiable?
$\neg F \approx S$

Definition (Ordered Resolution)

$$\frac{L \vee C \quad \neg L' \vee D}{(C \vee D)\sigma}$$

where

$L\sigma$ strictly maximal in $C\sigma$, $\neg L'\sigma$ maximal in $D\sigma$, $\sigma = \mathrm{mgu}(L, L')$.

Definition (Inst-Gen)

$$\frac{L \vee C \quad \neg L' \vee D}{(L \vee C)\sigma \quad (\neg L' \vee D)\sigma}$$

where

$$\mathrm{sel}(L \vee C) = L \qquad \mathrm{sel}(\neg L' \vee D) = \neg L' \qquad \sigma = \mathrm{mgu}(L, L')$$

Example (Resolution)

$$\cfrac{\cfrac{\mathsf{P}(x) \vee \neg\mathsf{P}(y) \quad \neg\mathsf{P}(\mathsf{a})}{\neg\mathsf{P}(y) \quad \mathsf{P}(\mathsf{b})}\; x \mapsto \mathsf{a}}{\square}\; y \mapsto \mathsf{b}$$

Example (Inst-Gen)

$$S_0\bot = \{\mathsf{P}(\bot) \vee \neg\mathsf{P}(\bot), \neg\mathsf{P}(\mathsf{a}), \mathsf{P}(\mathsf{b})\} \qquad \text{satisfiable}$$

$$\frac{\mathsf{P}(x) \vee \neg\mathsf{P}(y)) \quad \neg\mathsf{P}(\mathsf{a})}{\mathsf{P}(\mathsf{a}) \vee \neg\mathsf{P}(y)}\; x \mapsto \mathsf{a}$$

$$S_1\bot \supsetneq \{\neg\mathsf{P}(\mathsf{a}), \mathsf{P}(\mathsf{b}), \mathsf{P}(\mathsf{a}) \vee \neg\mathsf{P}(\bot)\} \qquad \text{satisfiable}$$

$$\frac{\mathsf{P}(\mathsf{b}) \quad \mathsf{P}(\mathsf{a}) \vee \neg\mathsf{P}(y)}{\mathsf{P}(\mathsf{a}) \vee \mathsf{P}(\mathsf{b})}\; y \mapsto \mathsf{b}$$

$$S_2\bot \supsetneq \{\neg\mathsf{P}(\mathsf{a}), \mathsf{P}(\mathsf{b}), \mathsf{P}(\mathsf{a}) \vee \neg\mathsf{P}(\mathsf{b})\} \qquad \text{unsatisfiable}$$

Subsumption

$S = \{C, D, \ldots\}$ $\exists\theta\ C\theta \subseteq D$ C subsumes D

S satisfiable $\overset{\checkmark}{\iff}$ $(S \setminus D)$ satisfiable

θ is proper, $S\bot$ satisfiable $\overset{\times}{\iff}$ $(S \setminus D)\bot$ satisfiable

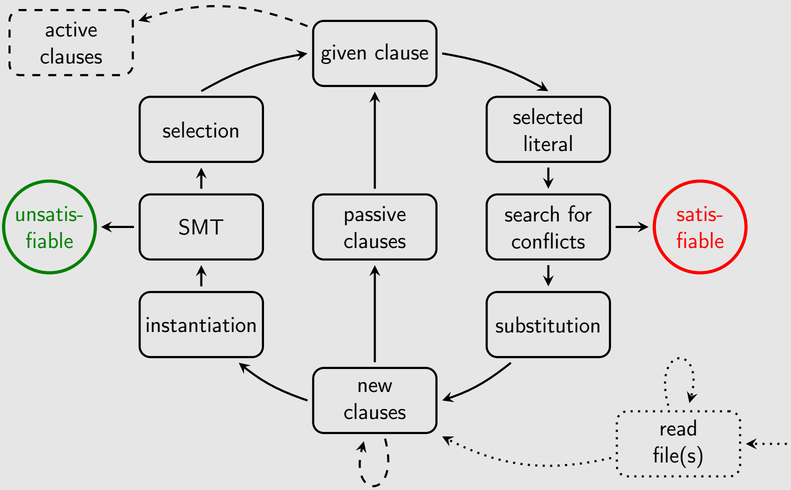
θ is renaming, $S\bot$ satisfiable $\overset{\checkmark}{\iff}$ $(S \setminus D)\bot$ satisfiable

Example

$\{\mathsf{P}(x, y), \neg\mathsf{P}(\mathsf{a}, z)\}$ $\{\mathsf{P}(x, y), \neg\mathsf{P}(\mathsf{a}, z), \mathsf{P}(\mathsf{a}, z)\}$

$\{\mathsf{P}(\bot, \bot), \neg\mathsf{P}(\mathsf{a}, \bot)\}$ $\{\mathsf{P}(\bot, \bot), \neg\mathsf{P}(\mathsf{a}, \bot), \mathsf{P}(\mathsf{a}, \bot)\}$

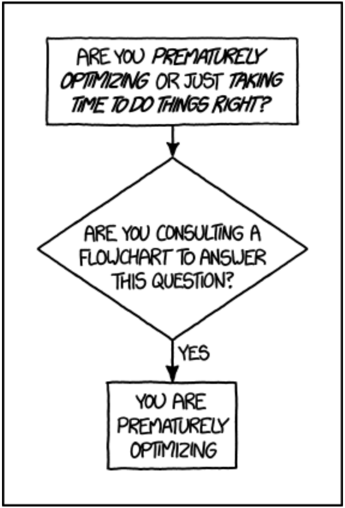
Run-Loop


active clauses
given clause
selection
selected literal
unsatis-fiable
SMT
passive clauses
search for conflicts
satis-fiable
instantiation
substitution
new clauses
read file(s)

```
func application(symbol:String, args:[term_t], tau:type_t) -> term_t {
guard args.count > 0 else { return constant(symbol, tau) }
let domain = [type_t](count:count, repeatedValue: free_tau)
let f = function(symbol, domain:domain(args.count, tau:Yices.free_tau),
range: term_tau)
```

1. Path indices of selected literals for clauses to find clashing literals
2. Discrimination trees of literals for clauses to find variants of clauses

- migrate to Linux / Swift 3
- integrate unit superposition
- integrate ordered maximal completion
- experiments
- optimizations


ARE YOU PREMATURELY OPTIMIZING OR JUST TAKING TIME TO DO THINGS RIGHT?
ARE YOU CONSULTING A FLOWCHART TO ANSWER THIS QUESTION?
YES
YOU ARE PREMATURELY OPTIMIZING