

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 Procedure
- 3 Equality
- 4 Bernays–Schönfinkel–Ramsey
- 5 work to do

References

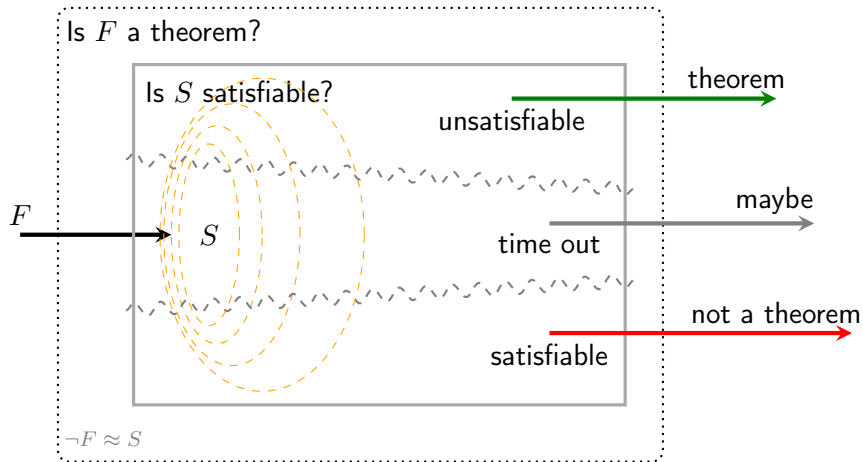


Bruno Dutertre, *Yices 2.2*, Computer-Aided Verification (CAV'2014) (Armin Biere and Roderick Bloem, eds.), Lecture Notes in Computer Science, vol. 8559, Springer, July 2014, pp. 737–744.



G. Sutcliffe, *The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0*, Journal of Automated Reasoning **43** (2009), no. 4, 337–362.

Goal



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 $D\sigma$, $\sigma = \text{mgu}(L, L')$.

Definition (Inst-Gen)

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

where

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

Example (Resolution)

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

Example (Inst-Gen)

$$S_0 \perp = \{P(\perp) \vee \neg P(\perp), \neg P(a), P(b)\} \quad \text{satisfiable}$$

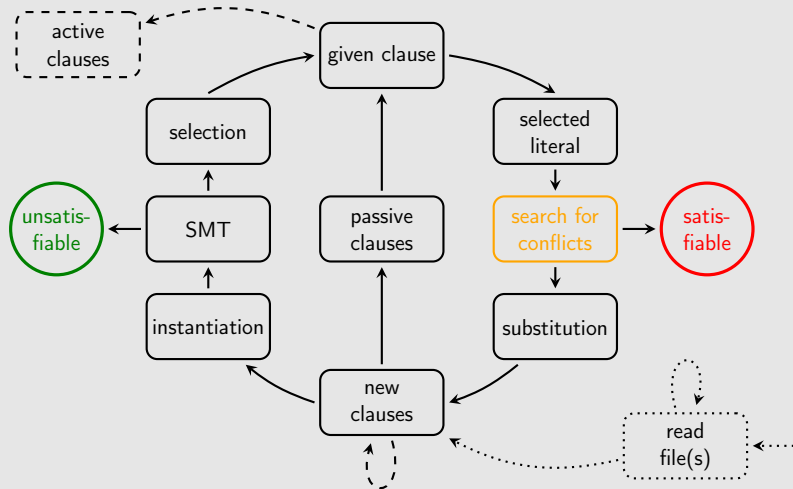
$$\frac{P(x) \vee \neg P(y) \quad \neg P(a)}{P(a) \vee \neg P(y)} \quad x \mapsto a$$

$$S_1 \perp \supsetneq \{\neg P(a), P(b), P(a) \vee \neg P(\perp)\} \quad \text{satisfiable}$$

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

$$S_2 \perp \supsetneq \{\neg P(a), P(b), P(a) \vee \neg P(b)\} \quad \text{unsatisfiable}$$

Run-Loop



Subsumption

$$S = \{C, D, \dots\} \quad \exists \theta \ C\theta \subseteq D \quad \text{C subsumes D}$$

$$S \text{ satisfiable} \xrightarrow{\checkmark} (S \setminus D) \text{ satisfiable}$$

$$\theta \text{ is proper, } S \perp \text{ satisfiable} \xrightarrow{\times} (S \setminus D) \perp \text{ satisfiable}$$

$$\theta \text{ is renaming, } S \perp \text{ satisfiable} \xrightarrow{\checkmark} (S \setminus D) \perp \text{ satisfiable}$$

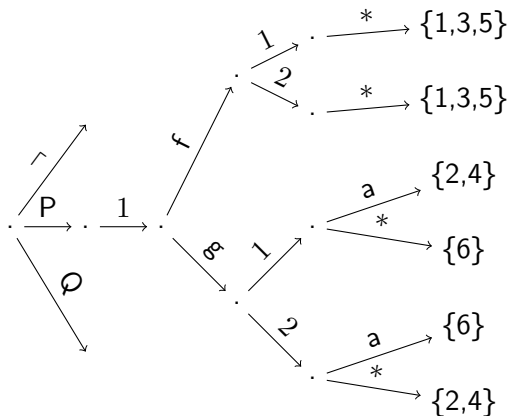
Example

$$\begin{array}{ll} \{P(x, y), \neg P(a, z)\} & \{P(x, y), \neg P(a, z), P(a, z)\} \\ \{P(\perp, \perp), \neg P(a, \perp)\} & \{P(\perp, \perp), \neg P(a, \perp), P(a, \perp)\} \end{array}$$

needed data structures

- 1 representation of clauses, literals and terms
- 2 fast retrieval of clauses with a selected literal that is unifiable with the negation selected literal of a given clause
- 3 fast retrieval of clauses with a set of literals that is a renamed subset of a given clause

$$\{^1: P(f(x, x)), ^2: P(g(a, x)), ^3: P(f(y, z)), ^4: P(g(a, y)), ^5: P(f(y, x)), ^6: P(g(y, a))\}$$

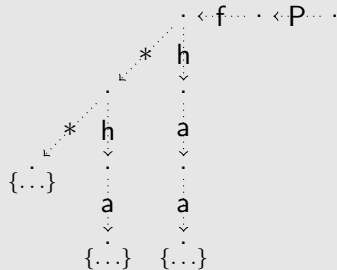


$$\neg P(g(b, z)) \mapsto \{P.1.g.1.b, P.1.g.2.*\} \mapsto \{6\} \cap \{2, 4, 6\}$$

$$\ell_1: P(f(x, y)), \ell_2: P(f(x, h(a))), \ell_3: P(f(h(a), a))$$

$$\ell_1 \mapsto P.f.*.*$$

$$\ell_2 \mapsto P.f.*.h.a$$

$$\ell_3 \mapsto P.f.h.a.a$$


Implementation

$$\text{Clause} \mapsto (\text{Int}, \text{Set of term_t})$$

$$\text{term_t} \mapsto \text{Set of Int}$$

$$\text{Int} \mapsto \text{Clause}$$

```
type_t yices_bool_type(void);  
type_t yices_new_uninterpreted_type(void);  
type_t yices_function_type(uint32_t n, const type_t dom[], type_t range);  
term_t yices_new_uninterpreted_term(type_t tau);  
term_t yices_application(term_t fun, uint32_t n, const term_t arg[]);  
term_t yices_eq(term_t left, term_t right);  
  
term_t yices_not(term_t arg);  
term_t yices_or(uint32_t n, term_t arg[]);  
  
int32_t yices_assert_formula(context_t *ctx, term_t t);  
  
smt_status_t yices_check_context(context_t *ctx, const param_t *params);  
model_t *yices_get_model(context_t *ctx, int32_t keep_subst);  
int32_t yices_get_bool_value(model_t *mdl, term_t t, int32_t *val);
```

Example

$S = \{\mathbf{P(a)}, \neg P(f(a, b)), f(x, b) = x\}$	unsatisfiable
$S \perp = \{\mathbf{P(a)}, \neg P(f(a, b)), f(\perp, b) = \perp\}$	satisfiable
$a \neq y \vee \neg P(a) \vee P(y)$	$P(a)$, congruence

Schemata

$x = x$	$s \neq s$
$x \neq y \vee y = x$	$s \neq t$
$x \neq y \vee y \neq z \vee x = z$	$s \neq t$
$x_1 \neq y_1 \vee x_2 \neq y_2 \vee \mathbf{f(x_1, x_2) = f(y_1, y_2)}$	$f(s_1, s_2) \neq f(t_1, t_2)$
$x \neq y \vee \neg \mathbf{P(x)} \vee P(y)$	$P(s)$
$x \neq y \vee \neg P(x) \vee \mathbf{P(y)}$	$\neg P(s)$

Lemma

Symmetry and transitivity are consequences of reflexivity and congruence.

Symmetry.

$$\begin{array}{c}
 \frac{x_1 \neq y_1 \vee x_2 \neq y_2 \vee x_1 \neq x_2 \vee y_1 = y_2}{x \neq y \vee x \neq x \vee x \neq x \vee y = x} \text{C} \\
 \frac{x \neq y \vee x \neq x \vee x \neq x \vee y = x}{x \neq y \vee y = x} \text{R}
 \end{array}
 \quad
 \begin{array}{l}
 x_1 \mapsto x, x_2 \mapsto x, y_1 \mapsto y, y_2 \mapsto x
 \end{array}$$



Transitivity.

$$\begin{array}{c}
 \frac{x_1 \neq y_1 \vee x_2 \neq y_2 \vee x_1 \neq x_2 \vee y_1 = y_2}{x \neq x \vee y \neq z \vee x \neq y \vee x = z} \text{C} \\
 \frac{x \neq x \vee y \neq z \vee x \neq y \vee x = z}{x \neq y \vee y \neq z \vee x = z} \text{R}
 \end{array}
 \quad
 \begin{array}{l}
 x_1 \mapsto x, x_2 \mapsto y, y_1 \mapsto x, y_2 \mapsto z
 \end{array}$$



The Bernays–Schönfinkel–Ramsey class of first-order formulae is a decidable fragment of first-order logic. Each formula in this fragment is equivalent to a satisfiable formula

$$\exists a_1 \dots a_m \forall y_1 \dots y_n F$$

where F is quantifier free and does not contain function symbols.

Example

$$a \neq x \vee Q(a, x), Q(y, b)$$

Remark

In practice the addition of equality axioms reduces the success rates of (instantiation-based) automated theorem provers drastically (even in this segment).

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

