ICS stands for Integrated Canonizer and Solver.

Core ICS is a Decision Procedure for a combination of theories

- Theory of Equality over uninterpreted function
- Theory of Integer and Rational Linear Arithmetic
- Theory of fixed length bit vector
- Theory of Tuples and Arrays
- Theory of propositional sets
- Theory of Coproducts and power products

Examples

\[
x + 2 = y, \; f(a[x := 3][y-2]) = f(y-x+1) + 3z = f(x-y), \; x = z + y, \; -y < -x-f(fz)\]
Other Features

- Decision Procedure for SAT through lazy cooperation of a non clausal SAT solver and the core ICS

- API(C,Lisp,Ocaml) for embedded deduction
  - suitable for symbolic simulation or proof search
  - online, multi-threaded

- Incomplete Extension nonlinear arithmetic and integer

- It is a really fast (30000 Theorems a second) see CAV’04
A Theory

A Signature $\Sigma = (\Sigma^C, \Sigma^F, \Sigma^P)$ where
$\Sigma^C$ = a set of constants
$\Sigma^F$ = a set of functions
$\Sigma^P$ = a set of predicates

$\Sigma_{\text{term}} := f(t_1, \ldots, t_n) \mid c \mid \text{variables}$
$\Sigma_{\text{atom}} := P(t_1, \ldots, t_n) \mid s = t$
where $P \in \Sigma^P, f \in \Sigma^F, c \in \Sigma^C$
t_1, \ldots, t_n, s, t are terms

\[\Sigma_{\text{formulae}} := \Sigma_{\text{atom}}\]
\[\|\Sigma_{\text{formulae}} \rightarrow \Sigma_{\text{formulae}}\]
\[\neg \Sigma_{\text{formulae}}\]
\[\Sigma_{\text{formulae}} \lor \Sigma_{\text{formulae}}\]
\[\Sigma_{\text{formulae}} \land \Sigma_{\text{formulae}}\]
\[\exists x \Sigma_{\text{formulae}}(x)\]
\[\forall x \Sigma_{\text{formulae}}(x)\]

$\Sigma_{\text{sentences}}$ are $\Sigma_{\text{formulae}}$ with no free variables.
Let $\Sigma$ be a signature.

A $\Sigma_{\text{interpretation}}$ $\mathcal{A}$ with domain $A$ is a map for each variable $x \in \text{variable}$ as an element $x^A \in A$ for each constants $c \in \Sigma^C$ s an element $c^A \in A$ for each function $f \in \Sigma^F$ of arity $n$ as a $f^A : A^n \to A$ for each function $P \in \Sigma^P$ of arity $n$ as a subset $P^A$ subset of $A^n$

A $\Sigma_{\text{Theory}}$ is a set of $\Sigma_{\text{sentences}}$ that are true in the interpretation $\mathcal{A}$ whose domain is $A$.

Given a Theory $T$, a formula $\varphi$ is

$T_{\text{valid}}$, if it evaluates to true under all interpretations over a domain

$T_{\text{satisfiable}}$, if it evaluates to true under some interpretations over a domain

$T_{\text{unsatisfiable}}$, if it evaluates to false under all interpretations over a domain
The Theory $T_Z$ of integers
Signature $\Sigma_Z = (C, +, -, \leq)$
$C = \{ c_n \mid \text{for each integer } n \}$
+ a binary function symbol,
- unary function symbol,
$\leq$ is a binary predicate
$T_Z$ is the set of $\Sigma_{sentences}$ that are true in the interpretation $\mathcal{A}$ over $\mathbb{Z}$. $T_Z$ is decidable and proved by Presburger through quantifier elimination algorithm and it is in NP-Complete problem.
If we add $\ast$ to $T_Z$ is undecidable.

The Theory $T_R$ of Reals.
Signature $\Sigma_R = (C, +, -, \leq)$
$C = \{ c_r \mid \text{for each rational } r \}$
$T_R$ is the set of $\Sigma_{sentences}$ that are true in the interpretation $\mathcal{A}$ over $\mathbb{R}$.
$T_R$ is decidable through quantifier elimination algorithm
If we add $\ast$ to $T_R$ is decidable and proved by Tarski
The Theory $T_L$ of lists
Signature $\Sigma_L = (\text{cons}, \text{car}, \text{cdr})$
$\text{cons}$ is a binary symbol. $\text{car}, \text{cdr}$ is a unary symbol.
construction axiom
$\text{cons}(\text{car}(x), \text{cdr}(x)) = x$.
selection axiom
$\text{car}(\text{cons}(x, y)) = x, \text{cdr}(\text{cons}(x, y)) = y$,
an finite number of acyclicity axiom
$\text{car}(x) \neq x, \text{cdr}(x) \neq x, \text{car}(\text{car}(x)) \neq x, \text{car}(\text{cdr}(x)) \neq x$
quantifier free $T_L$ formula is decided in linear time.

The Theory $T_A$ of lists
Signature $\Sigma_L = (\text{read}, \text{write})$
$\text{read}$ is a binary functionsymbol. $\text{write}$ is a ternary function symbol.
construction axiom
$\text{read}(\text{write}(a, i, e), i) = e, i <> j \rightarrow \text{read}(\text{write}(a, i, e), j) = \text{read}(a, j)$.
$T_A$ is undecidable but quantifier free $T_A$ is decidable.
The Theory $T_E$ of Equality with uninterpreted function
Signature $\Sigma_E = (NULL, F, NULL)$ Axioms

- $x=y \Rightarrow y=x$
- $x=y \text{ AND } y=z \Rightarrow x=z$
- $x=y \Rightarrow f(x)=f(y)$ where $f$ is uninterpreted function

$T_E$ is undecidable but quantifier free $T_E$ is decidable based on congruence closure.
PROBLEM

Given the above theories
\[ T = T_E + T_A + T_L + T_B + T_Z + T_R \]
and a formula \( \varphi \).
Is quantifier free formula \( \varphi \) valid or satisfiable in \( T \)?

There are two basis approach
1. Nelson Oppen(NO) Method
Given \( n \) theories \( T_1, \ldots, T_N \), NO method combine the available decision procedure for \( T_1, \ldots, T_N \) into a single decision procedure for the satisfiability of quantifier free formula.
2. Shostak Method
Combines theory of equality \( T_E \) over uninterpreted function with several disjoint shostak theories \( T_1, T_2, \ldots, T_N \).
A shostak theory is a theory that can be canonizable and solvable.
A canonizer \( \rho \) maps terms to normal form terms s.t. equal terms in the theory are mapped to same form.
A solver maps an equation to an “Equation solved form”
A theory $T$ is canonizable if there is a canonizer $\rho$

1. $|\models_T a=b \text{ iff } \rho(a) \equiv \rho(b)$
2. $\rho(x) \equiv x$.
3. $\text{var}(\rho(a))$ is a sub set of $\rho(a)$.
4. $\rho(\rho(a)) \equiv \rho(a)$
5. If $\rho(a) \equiv f(b_1, \ldots, b_n)$, then $\rho(b_i) \equiv b_i$ for $0 < i < n+1$.

Examples

$\rho(y + x + 3 + x) = y + 2x + 3$

syntax for canonizer in ICS

*can term*
A theory $T$ is solvable if there is a solver $solve(a = b)$ s.t.
if $s \not\models t$ then $solve(s=t) = false$
if $s \models t$ then $solve(s=t) = true$
otherwise $solve$ returns a substitution

$$\rho = \{x_1 \leftarrow t_1, x_2 \leftarrow t_2, \ldots, x_n \leftarrow t_n\}$$

$s_i \in vars(s = t)$ and $s_i \not\in vars(t_j)$ forall $i,j$

syntax for solver in ICS

```plaintext
can th term= term
```

Examples

$solve la x + 2 = y - 3.$
$solve p car(x) = cons(u, v).$
A satisfiable solver for propositional formula over atoms.
$sat < formula >.$
$satp&[p|r].$
$sat[x = 1|x = 2|x = 3] & x > 1.$
**Solution for Decision problem**

Key idea
Maintain Theory-wise solution set
Communicate variable equalities as in NO
Construct combined canonizer (as required in a shostak algorithm).

Asserting facts
syntax in ICS for inserting fact
assert [@<ident>] <atom> <atom>
adds the atoms to the current context.

Return unsat, valid or ok
assert 2 * car(x) - 3* cdr(x) = f(cdr(x)).

show. syntax to set the theory wise partition
show

**New Application**

Infinite BMC Full ICS allows BMC to be extended from hardware consisting of purely Boolean circuits to software/systems whose states are defined over integers.