A T-Solver for EUF

Albert Oliveras and Enric Rodríguez-Carbonell

Deduction and Verification Techniques
Session 2
Fall 2009, Barcelona

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

The EUF logic

Equality with Uninterpreted Functions: (Burch and Dill, 1994) The syntax is the following one:

$$
\begin{aligned}
formula :== \quad & \textbf{true} \mid \textbf{false} \mid predicateSymbol(term, \cdots, term) \\
& \mid \neg formula \mid (formula \vee formula) \mid (formula \wedge formula) \mid \\
& \mid (term = term)
\end{aligned}
$$

$$
\begin{aligned}
term :== \quad & functionSymbol(term, \cdots, term) \\
& \mid ite(formula, term, term)
\end{aligned}
$$

Roughly speaking, ground first-order formulae with equality

The EUF logic EUF (2)

Ground first-order formulae with equality

Example 1: $a \neq c \quad \vee \quad b \neq d \quad \vee \quad f(a,b) = f(c,d)$
is valid (i.e. tautology)

Example 2: $f(f(f(a))) \neq b \ \wedge \ f(a) = a \ \wedge \ a = b$
is unsatisfiable

Example 3: $(\ P(a) \wedge \neg P(b)\ ) \ \vee \ a \neq b$
is satisfiable, but $a = b$ falsifies it

Deciding satisfiability NP-complete.

The EUF logic (3)

Applications:
– Processor verification (Dill, Bryant et al.)
– (Finite) model finding in FOL
for consistency proofs, inductive theorem proving, CSP's ...

Example: there exist groups of card. 4 iff S is satisfiable:

Group axioms:

$$\begin{aligned} f(e,x) &= x \\ f(i(x),x) &= e \\ f(f(x,y),z) &= f(x,f(y,z)) \end{aligned}$$

S has 4 new cts. a,b,c,d:

$a \neq b \wedge \ldots \wedge c \neq d$
$f(e,a) = a \wedge \ldots \wedge f(e,d) = d$
$f(i(a),a) = e \wedge \ldots$
$\ldots$
$e = a \vee e = b \vee e = c \vee e = d$
$f(a,a) = a \vee f(a,a) = b \ldots$
$\ldots$

EUF: current methods

Translate to propositional SAT and use DPLL:

–Bryant,German,Velev [ACM TOCL'01]
–MACE2 (McCune 1995)
–DDPP (Stickel 1994)

Specific techniques for finding FO models:

–Finder,SEM (Zhang,Zhang, 1995), Falcon (Zhang 96)
–MGTP (Hasegawa et al, 1992)
–MACE4 (McCune 2002)

Specific techniques for more general logics:

–Lemmas on Demand (de Moura, Ruess 2002)

Our approach: DPLL(T)

- DPLL(T) = DPLL(X) + T-solver
- DPLL(X) communicates with T-solver via an API
- Example:
  - `abstractLiteral(Term t, Var propVar)` returns `Lit`
  - `setTrue(Lit lit)`
  - `checkConsistency( )` returns `SAT/UNSAT`
  - `extractInconsistency( )` returns `Set of Lit`
  - `theoryPropagate( )` returns `Set of Lit`
  - `explain(Lit lit)` returns `Set of Lit`
  - `setBacktrackPoint( )`
  - `backtrack(int n)`

Congruence closure

The problem: deduction in ground equational theories

Example:

$$\left.\begin{array}{rcl} f(a,g(a)) & = & g(b) \\ g(a) & = & h(a) \\ a & = & f(c,h(c)) \\ h(a) & = & a \\ c & = & h(h(h(a))) \end{array}\right\} \models \quad a = g(b) \ ?$$

- Decidable, Ackerman 1954
- $O(n \ log \ n)$ Downey,Sethi,Tarjan 1980 JACM
- See also: Kozen STOC'77,
  Nelson,Oppen JACM'80, Shostak JACM'84
- Many applications:
  –compilers (common subexpresions),
  –verification, deduction (combination of theories, ...)

Our approach for EUF: $DPLL(=)$

- Unit propagation: many calls to congruence closure (CC)
- $O(n\ log\ n)$ algorithm of Downey,Sethi,Tarjan:
  - requires initial transformations to graph of outdegree 2
  - heavily relies on pointers and sharing
  - not as clean as later abstract versions of CC: [Kapur97, BachmairTiwariVigneron00] (generally $O(n^2)$).
- Ground completion algorithms are $O(n^2)$ [PS96] or rely on classical $O(n\ log\ n)$ CC-algorithms [Snyder89]
- Our approach is $O(n\ log\ n)$ but clean and simple.
- Idea: two initial transformations at the formula level done in the $DPLL(=)$ framework once and for all on the initial EUF problem (not at each call to CC).

The two initial transformations:

1. Curryfy (like in the implementation of FP):
   - After Curryfying: only one binary symbol "$\cdot$" and constants.
   - Example: Curryfying $f(a, g(b), c)$ gives $\cdot(\cdot(\cdot(f, a), \cdot(g, b)), c)$

2. Flatten:
   - Allows one to assume: terms of depth ≤ 1
   - Introduces a linear number of new constants
   - Example: Flattening $\{\ \cdot(\cdot(\cdot(f, a), \cdot(g, b)), c) \ = i\}$ gives
     $\{\ \cdot(f, a) = d,\ \ \cdot(g, b) = e,\ \ \cdot(d, e) = h,\ \ \cdot(h, c) = i\ \}$

After this:

- Literals in EUF formula between cts. only: $a = b$ or $a \neq b$
- Hidden inside the CC module there is a fixed set of equations E of the form $\cdot(a, b) = c$

Congruence closure: our view

Now the CC problem is: $E \models a = b$? $(a, b, c, d, e$ cts.)
where in E there are only equations of the form $\cdot(c, d) = e$

Our data structures: (no union-find!)

1. Pending unions: a list of pairs of cts yet to be merged.
2. Representative table: array indexed by constants, with for each constant c its current representative $rep(c)$.
3. Class lists: for each repres., the list of all cts in its class.
4. Lookup table: for each input term $\cdot(a, b)$, $Lookup(rep(a), rep(b))$ returns in constant time a constant c such that $\cdot(a, b) = c$ ($\perp$ if there is none).
5. Use lists: for each representative a, the list of input equations $\cdot(b, c) = d$ such that a is $rep(b)$ or $rep(c)$ or both.

Congruence closure: our algorithm

Notation: c' means $rep(c)$

```
While Pending ≠ ∅ Do
  remove a = b from Pending
  If a' ≠ b' and, wlog., |ClassList(a')| ≤ |ClassList(b')| Then
    For each c in ClassList(a') Do
      set rep(c) to b' and add c to ClassList(b')
    EndFor
    For each ·(c,d) = e in UseList(a') Do
      If Lookup(c',d') is some f and f' ≠ e' Then
        add e' = f' to Pending
      EndIf
      set Lookup(c',d') to e'
      add ·(c,d) = e to UseList(b')
    EndFor
  EndIf
EndWhile
```

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Congruence closure: our algorithm (2)

Notation: c' means $rep(c)$

While $Pending \neq \emptyset$ Do
  remove $a = b$ from $Pending$
  If $a' \neq b'$ and, wlog., $|ClassList(a')| \leq |ClassList(b')|$ Then
    For each c in $ClassList(a')$ Do
      set $rep(c)$ to b' and add c to $ClassList(b')$
    EndFor
    For each $\cdot(c,d) = e$ in $UseList(a')$ Do
      If $Lookup(c',d')$ is some f and $f' \neq e'$ Then
        add $e' = f'$ to $Pending$
      EndIf
      set $Lookup(c',d')$ to e'
      add $\cdot(c,d) = e$ to $UseList(b')$
    EndFor
  EndIf
EndWhile

$a' = b'$

$\cdot(a',d') = e \quad \cdot(b',d') = f$

Running example

$$\left.\begin{array}{rcl} f(a) & = & g(b) \\ g(c) & = & h(f(c),g(a)) \\ b & = & c \\ f(c) & = & g(a) \\ h(d,d) & = & g(b) \\ g(a) & = & d \end{array}\right\} \Longrightarrow \left[\begin{array}{rcl} \cdot(f,a) & = & e_1 \\ \cdot(g,b) & = & e_2 \\ \cdot(g,c) & = & e_3 \\ \cdot(f,c) & = & e_4 \\ \cdot(h,e_4) & = & e_5 \\ \cdot(g,a) & = & e_6 \\ \cdot(e_5,e_6) & = & e_7 \\ \cdot(h,d) & = & e_8 \\ \cdot(e_8,d) & = & e_9 \end{array}\right] + \left[\begin{array}{rcl} e_1 & = & e_2 \\ e_3 & = & e_7 \\ b & = & c \\ e_4 & = & e_6 \\ e_9 & = & e_2 \\ e_6 & = & d \end{array}\right]$$

And we initialize *Lookup* table:

$$\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\}$$

Running example (contd.)

$$\left.\begin{array}{rcl} f(a) & = & g(b) \\ g(c) & = & h(f(c),g(a)) \\ b & = & c \\ f(c) & = & g(a) \\ h(d,d) & = & g(b) \\ g(a) & = & d \end{array}\right\} \Longrightarrow \left[\begin{array}{rcl} \cdot(f,a) & = & e_1 \\ \cdot(g,b) & = & e_2 \\ \cdot(g,c) & = & e_3 \\ \cdot(f,c) & = & e_4 \\ \cdot(h,e_4) & = & e_5 \\ \cdot(g,a) & = & e_6 \\ \cdot(e_5,e_6) & = & e_7 \\ \cdot(h,d) & = & e_8 \\ \cdot(e_8,d) & = & e_9 \end{array}\right] + \left[\begin{array}{rcl} e_1 & = & e_2 \\ e_3 & = & e_7 \\ b & = & c \\ e_4 & = & e_6 \\ e_9 & = & e_2 \\ e_6 & = & d \end{array}\right]$$

Similarly, initialization for *UseList* is:

$$UseList(a) = \{\cdot(f,a) = e_1, \cdot(g,a) = e_6\}$$
$$UseList(b) = \{\cdot(g,b) = e_2\}$$
$$UseList(c) = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4\}$$

Running example (contd.)

$$\begin{bmatrix} \textit{Pending} \\ e_1 = e_2 \\ e_3 = e_7 \\ b = c \\ e_4 = e_6 \\ e_9 = e_2 \\ e_6 = d \end{bmatrix} \begin{bmatrix} \textit{UseList} & \\ b = \{\cdot(g,b) = e_2\} & e_6 = \{\cdot(e_5,e_6) = e_7\} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4\} & e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} & e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_4 = \{\cdot(h,e_4) = e_5\} & e_5 = \{\cdot(e_5,e_6) = e_7\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\}$

Running example (contd.)

In normal form: $e_1 \rightarrow e_2$.

$$\begin{bmatrix} \textit{Pending} \\ e_1 = e_2 \\ e_3 = e_7 \\ b = c \\ e_4 = e_6 \\ e_9 = e_2 \\ e_6 = d \end{bmatrix} \begin{bmatrix} \textit{UseList} & \\ b = \{\cdot(g,b) = e_2\} & e_6 = \{\cdot(e_5,e_6) = e_7\} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4\} & e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} & e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_4 = \{\cdot(h,e_4) = e_5\} & e_5 = \{\cdot(e_5,e_6) = e_7\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\}$

Running example (contd.)

In normal form: $e_1 \rightarrow e_2$. We have $UseList(e_1) = \emptyset$.

$$\begin{bmatrix} \textit{Pending} \\ e_1 = e_2 \\ e_3 = e_7 \\ b = c \\ e_4 = e_6 \\ e_9 = e_2 \\ e_6 = d \end{bmatrix} \begin{bmatrix} \textit{UseList} & \\ b = \{\cdot(g,b) = e_2\} & e_6 = \{\cdot(e_5,e_6) = e_7\} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4\} & e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} & e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_4 = \{\cdot(h,e_4) = e_5\} & e_5 = \{\cdot(e_5,e_6) = e_7\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	b	c	d	e_2	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\}$

Running example (contd.)

In normal form: $e_3 \longrightarrow e_7$.

$$\begin{bmatrix} \textit{Pending} \\ e_3 = e_7 \\ b = c \\ e_4 = e_6 \\ e_9 = e_2 \\ e_6 = d \end{bmatrix} \begin{bmatrix} \textit{UseList} \\ b = \{\cdot(g,b) = e_2\} & e_6 = \{\cdot(e_5,e_6) = e_7\} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4\} & e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} & e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_4 = \{\cdot(h,e_4) = e_5\} & e_5 = \{\cdot(e_5,e_6) = e_7\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	b	c	d	e_2	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\}$

Running example (contd.)

In normal form: $e_3 \rightarrow e_7$. We have $UseList(e_3) = \emptyset$.

$$\begin{bmatrix} \textit{Pending} \\ e_3 = e_7 \\ b = c \\ e_4 = e_6 \\ e_9 = e_2 \\ e_6 = d \end{bmatrix} \begin{bmatrix} \textit{UseList} \\ b = \{\cdot(g,b) = e_2\} & e_6 = \{\cdot(e_5,e_6) = e_7\} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4\} & e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} & e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_4 = \{\cdot(h,e_4) = e_5\} & e_5 = \{\cdot(e_5,e_6) = e_7\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	b	c	d	e_2	e_2	e_7	e_4	e_5	e_6	e_7	e_8	e_9

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\}$

Running example (contd.)

In normal form: $b \rightarrow c$.

$$\begin{bmatrix} \textit{Pending} \\ b = c \\ e_4 = e_6 \\ e_9 = e_2 \\ e_6 = d \end{bmatrix} \begin{bmatrix} \textit{UseList} & \\ b = \{\cdot(g,b) = e_2\} & e_6 = \{\cdot(e_5,e_6) = e_7\} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4\} & e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} & e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_4 = \{\cdot(h,e_4) = e_5\} & e_5 = \{\cdot(e_5,e_6) = e_7\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	d	e_2	e_2	e_7	e_4	e_5	e_6	e_7	e_8	e_9

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\}$

Running example (contd.)

In normal form: $b \rightarrow c$. Let's treat $\cdot(g,b) = e_2$. Since $Lookup(g',b') = Lookup(g,c) = e_3$ and $e'_3 \neq e'_2$, we add $e_7 = e_2$ to *Pending*.

$$\left[\begin{array}{ccc} \multicolumn{3}{c}{\textit{Pending}} \\ b & = & c \\ e_4 & = & e_6 \\ e_9 & = & e_2 \\ e_6 & = & d \end{array}\right] \left[\begin{array}{ll} \multicolumn{2}{c}{\textit{UseList}} \\ b = \{\cdot(g,b) = e_2\} & e_6 = \{\cdot(e_5,e_6) = e_7\} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4\} & e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} & e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_4 = \{\cdot(h,e_4) = e_5\} & e_5 = \{\cdot(e_5,e_6) = e_7\} \end{array}\right]$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	d	e_2	e_2	e_7	e_4	e_5	e_6	e_7	e_8	e_9

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\}$

Running example (contd.)

In normal form: $b \rightarrow c$. Let's treat $\cdot(g,b) = e_2$. Since $Lookup(g',b') = Lookup(g,c) = e_3$ and $e'_3 \neq e'_2$, we add $e_7 = e_2$ to $Pending$. Now, $Lookup(g,c) = e_7$ and add $\cdot(g,b) = e_2$ to $UseList(c)$.

$$\left[\begin{array}{ccc} \multicolumn{3}{c}{Pending} \\ e_7 & = & e_2 \\ e_4 & = & e_6 \\ e_9 & = & e_2 \\ e_6 & = & d \end{array}\right] \left[\begin{array}{cc} \multicolumn{2}{c}{UseList} \\ \multicolumn{2}{c}{c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(g,b) = e_2\}} \\ e_6 = \{\cdot(e_5,e_6) = e_7\} & e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} & e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_4 = \{\cdot(h,e_4) = e_5\} & e_5 = \{\cdot(e_5,e_6) = e_7\} \end{array}\right]$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	d	e_2	e_2	e_7	e_4	e_5	e_6	e_7	e_8	e_9

$Lookup : \{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_7, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\}$

Running example (contd.)

In normal form: $e_2 \rightarrow e_7$.

$$\begin{bmatrix} \textit{Pending} \\ e_7 = e_2 \\ e_4 = e_6 \\ e_9 = e_2 \\ e_6 = d \end{bmatrix} \begin{bmatrix} \textit{UseList} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(g,b) = e_2\} \\ e_6 = \{\cdot(e_5,e_6) = e_7\} \quad e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} \quad e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_4 = \{\cdot(h,e_4) = e_5\} \quad e_5 = \{\cdot(e_5,e_6) = e_7\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	d	e_7	e_7	e_7	e_4	e_5	e_6	e_7	e_8	e_9

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_7, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\}$

Running example (contd.)

In normal form: $e_2 \rightarrow e_7$. Again $UseList(e_2) = \emptyset$.

$$\begin{bmatrix} \textit{Pending} \\ e_7 = e_2 \\ e_4 = e_6 \\ e_9 = e_2 \\ e_6 = d \end{bmatrix} \begin{bmatrix} \textit{UseList} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(g,b) = e_2\} \\ e_6 = \{\cdot(e_5,e_6) = e_7\} \quad e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} \quad e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_4 = \{\cdot(h,e_4) = e_5\} \quad e_5 = \{\cdot(e_5,e_6) = e_7\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	d	e_7	e_7	e_7	e_4	e_5	e_6	e_7	e_8	e_9

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_7, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\}$

Running example (contd.)

In normal form: $e_4 \longrightarrow e_6$.

$$\begin{bmatrix} \textit{Pending} \\ e_4 = e_6 \\ e_9 = e_2 \\ e_6 = d \end{bmatrix} \begin{bmatrix} \textit{UseList} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(g,b) = e_2\} \\ e_6 = \{\cdot(e_5,e_6) = e_7\} & e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} & e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_4 = \{\cdot(h,e_4) = e_5\} & e_5 = \{\cdot(e_5,e_6) = e_7\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	d	e_7	e_7	e_7	e_6	e_5	e_6	e_7	e_8	e_9

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_7, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\}$

Running example (contd.)

In normal form: $e_4 \rightarrow e_6$. Let's treat $\cdot(h, e_4) = e_5$. Since $Lookup(h', e'_4) = Lookup(h, e_6) = \emptyset$, just add $Lookup(h, e_6) = e_5$ to $Lookup$ and $\cdot(h, e_4)$ to $UseList(e_6)$.

$$\left[\begin{array}{ccc} \multicolumn{3}{c}{Pending} \\ e_4 & = & e_6 \\ e_9 & = & e_2 \\ e_6 & = & d \end{array}\right] \left[\begin{array}{cc} \multicolumn{2}{c}{UseList} \\ \multicolumn{2}{c}{c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(g,b) = e_2\}} \\ e_6 = \{\cdot(e_5, e_6) = e_7\} & e_8 = \{\cdot(e_8, d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} & e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_4 = \{\cdot(h, e_4) = e_5\} & e_5 = \{\cdot(e_5, e_6) = e_7\} \end{array}\right]$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	d	e_7	e_7	e_7	e_6	e_5	e_6	e_7	e_8	e_9

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_7, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\}$

Running example (contd.)

In normal form: $e_4 \rightarrow e_6$. Let's treat $\cdot(h, e_4) = e_5$. Since $Lookup(h', e'_4) = Lookup(h, e_6) = \emptyset$, just add $Lookup(h, e_6) = e_5$ to $Lookup$ and $\cdot(h, e_4)$ to $UseList(e_6)$.

$$\begin{bmatrix} \textit{Pending} \\ e_4 = e_6 \\ e_9 = e_2 \\ e_6 = d \end{bmatrix} \begin{bmatrix} \textit{UseList} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(g,b) = e_2\} \\ e_6 = \{\cdot(e_5,e_6) = e_7, \cdot(h,e_6) = e_5\} \quad e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} \quad e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_5 = \{\cdot(e_5,e_6) = e_7\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	d	e_7	e_7	e_7	e_6	e_5	e_6	e_7	e_8	e_9

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_7, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9, \cdot(h,e_6) = e_5\}$

Running example (contd.)

In normal form: $e_9 \rightarrow e_7$ (and not the other way around). Again $UseList(e_9) = \emptyset$.

$$\begin{bmatrix} \textit{Pending} \\ e_9 \quad = \quad e_2 \\ e_6 \quad = \quad d \end{bmatrix} \begin{bmatrix} \textit{UseList} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(g,b) = e_2\} \\ e_6 = \{\cdot(e_5,e_6) = e_7, \cdot(h,e_6) = e_5\} \quad e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} \quad e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_5 = \{\cdot(e_5,e_6) = e_7\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	d	e_7	e_7	e_7	e_6	e_5	e_6	e_7	e_8	e_7

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_7, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9, \cdot(h,e_6) = e_5\}$

Running example (contd.)

In normal form: $d \rightarrow e_6$ (and not the other way around).

$$\left[\begin{array}{c} \textit{Pending} \\ e_6 \quad = \quad d \end{array}\right] \left[\begin{array}{ll} \multicolumn{2}{c}{\textit{UseList}} \\ \multicolumn{2}{c}{c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(g,b) = e_2\}} \\ e_6 = \{\cdot(e_5,e_6) = e_7, \cdot(h,e_6) = e_5\} & e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} & e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_5 = \{\cdot(e_5,e_6) = e_7\} & \end{array}\right]$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	e_6	e_7	e_7	e_7	e_6	e_5	e_6	e_7	e_8	e_7

$$\textit{Lookup} : \{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_7, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9, \cdot(h,e_6) = e_5\}$$

Running example (contd.)

In normal form: $d \rightarrow e_6$ (and not the other way around). Let's treat $\cdot(h,d) = e_8$. Since $Lookup(h',d') = Lookup(h,e_6) = e_5$ and $e'_8 \neq e'_5$, add $e_5 = e_8$ to *Pending*. We also add $\cdot(h,d) = e_8$ to $UseList(e_6)$.

$$\begin{bmatrix} \textit{Pending} \\ e_6 \quad = \quad d \end{bmatrix} \begin{bmatrix} \textit{UseList} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(g,b) = e_2\} \\ e_6 = \{\cdot(e_5,e_6) = e_7, \cdot(h,e_6) = e_5\} \quad e_8 = \{\cdot(e_8,d) = e_9\} \\ d = \{\cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} \quad e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_5 = \{\cdot(e_5,e_6) = e_7\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	e_6	e_7	e_7	e_7	e_6	e_5	e_6	e_7	e_8	e_7

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_7, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9, \cdot(h,e_6) = e_5\}$

Running example (contd.)

In normal form: $d \rightarrow e_6$ (and not the other way around). Let's treat $\cdot(h,d) = e_8$. Since $Lookup(h',d') = Lookup(h,e_6) = e_5$ and $e'_8 \neq e'_5$, add $e_5 = e_8$ to *Pending*. We also add $\cdot(h,d) = e_8$ to $UseList(e_6)$.

$$\begin{bmatrix} \textit{Pending} \\ e_5 \quad = \quad e_8 \end{bmatrix} \begin{bmatrix} \textit{UseList} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(g,b) = e_2\} \\ e_6 = \{\cdot(e_5,e_6) = e_7, \cdot(h,e_6) = e_5, \cdot(h,d) = e_8\} \\ d = \{\cdot(h,d) = e_8\} \qquad e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_5 = \{\cdot(e_5,e_6) = e_7\} \quad e_8 = \{\cdot(e_8,d) = e_9\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	e_6	e_7	e_7	e_7	e_6	e_5	e_6	e_7	e_8	e_7

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_7, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9, \cdot(h,e_6) = e_5\}$

Running example (contd.)

In normal form: $d \rightarrow e_6$ (and not the other way around). Let's treat $\cdot(h,d) = e_8$. Since $Lookup(h',d') = Lookup(h,e_6) = e_5$ and $e'_8 \neq e'_5$, add $e_5 = e_8$ to *Pending*. We also add $\cdot(h,d) = e_8$ to $UseList(e_6)$.

$$\begin{bmatrix} \textit{Pending} \\ e_5 \quad = \quad e_8 \end{bmatrix} \begin{bmatrix} \textit{UseList} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(g,b) = e_2\} \\ e_6 = \{\cdot(e_5,e_6) = e_7, \cdot(h,e_6) = e_5, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} \\ e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_5 = \{\cdot(e_5,e_6) = e_7\} \quad e_8 = \{\cdot(e_8,d) = e_9\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	e_6	e_7	e_7	e_7	e_6	e_5	e_6	e_7	e_8	e_7

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_7, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9, \cdot(h,e_6) = e_5, \cdot(e_8,e_6) = e_9\}$

Running example (contd.)

In normal form: $e_5 \rightarrow e_8$. Let's treat $\cdot(e_5, e_6) = e_7$. Since $Lookup(e'_5, e'_6) = Lookup(e_8, e_6) = e_9$, we should add $e'_7 = e'_9$, but it is discarded because it already holds.

$$\begin{bmatrix} \textit{Pending} \\ e_5 \quad = \quad e_8 \end{bmatrix} \begin{bmatrix} \textit{UseList} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(g,b) = e_2\} \\ e_6 = \{\cdot(e_5,e_6) = e_7, \cdot(h,e_6) = e_5, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} \\ e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_5 = \{\cdot(e_5,e_6) = e_7\} \quad e_8 = \{\cdot(e_8,d) = e_9\} \end{bmatrix}$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	e_6	e_7	e_7	e_7	e_6	e_8	e_6	e_7	e_8	e_7

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_7, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9, \cdot(h,e_6) = e_5, \cdot(e_8,e_6) = e_9\}$

Running example (contd.)

Now, we could ask whether $g(a) = h(d,d)$ holds. After curryfing and flattening, the question is whether $e_6 = e_9$, which is false. On the other hand, we can check that $g(c) = h(f(b),d)$ because it is equivalent to $e_3 = e_9$, which is obviously true.

$$\left[\ \textit{Pending}\ \right] \left[\begin{array}{c} \textit{UseList} \\ c = \{\cdot(g,c) = e_3, \cdot(f,c) = e_4, \cdot(g,b) = e_2\} \\ e_6 = \{\cdot(e_5,e_6) = e_7, \cdot(h,e_6) = e_5, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9\} \\ e_1 = e_2 = e_3 = e_9 = \emptyset \\ e_5 = \{\cdot(e_5,e_6) = e_7\} \quad e_8 = \{\cdot(e_8,d) = e_9\} \end{array}\right]$$

Constant	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
Representative	a	c	c	e_6	e_7	e_7	e_7	e_6	e_8	e_6	e_7	e_8	e_7

Lookup : $\{\cdot(f,a) = e_1, \cdot(g,b) = e_2, \cdot(g,c) = e_7, \cdot(f,c) = e_4, \cdot(h,e_4) = e_5, \cdot(g,a) = e_6, \cdot(e_5,e_6) = e_7, \cdot(h,d) = e_8, \cdot(e_8,d) = e_9, \cdot(h,e_6) = e_5, \cdot(e_8,e_6) = e_9\}$

Analysis of the algorithm

$O(n\ log\ n)$ time and linear space:

- assume k different constants (usually, $k \ll n$)
- each ct changes representative at most $log\ k$ times
- maintenance *rep* and *ClassList*: $k\ log\ k$
- maintentance *Lookup* and *UseList*: $2n\ log\ k$

Correctness:

- Let *RepresentativeE* be the non-trivial eqs $a = a'$ and $\cdot(a', b') = c'$ where a, b and c cts in E_0 and c is $Lookup(a', b')$.
- Note: final *RepresentativeE* is the resulting closure (a convergent TRS)
- Key invariant: $(RepresentativeE \cup Pending)^* = E_0^*$

Experimental results

		Our algorithm	
File	**Abstract CC**	**No preprocess**	**Preprocess**
ex4211.5000	0.212547	0.039415	1.218427
ex4301.5000	3.720394	0.077519	1.409287
ex4301.6000	4.282850	0.092307	1.738292
ex4211.7000	0.270680	0.044061	1.641836
ex4301.7000	2.293164	0.107017	2.003430
ex4211.10000	0.357135	0.040921	2.365986
...			
TOTAL TIME	39.452168	0.738211	23.636892

CONCLUSION:Better performance, specially once preprocessed

Integer Offsets

- Bryant et al. add interpreted succ and pred symbols, extending EUF to CLU logic.
- The syntax is now the following one:

$$
\begin{aligned}
formula :== \quad & \textbf{true} \mid \textbf{false} \mid predicateSymbol(term, \cdots, term) \\
& \mid \neg formula \mid (formula \vee formula) \mid (formula \wedge formula \\
& \mid (term = term)
\end{aligned}
$$

$$
\begin{aligned}
int\_term :== \quad & functionSymbol(int\_term, \cdots, int\_term) \\
& \mid ite(formula, int\_term, int\_term) \\
& succ(int\_term) \mid pred(int\_term)
\end{aligned}
$$

- Note that all non-boolean terms are interpreted over the integers

Integer Offsets (contd.)

- write (sub)terms $\underbrace{succ(\ldots succ}_{k\ times}(t)\ldots)$ as $t+k$

  same with negative k for $\underbrace{pred(\ldots pred}_{k\ times}(t)\ldots)$

- Example: $f(a)=c \ \wedge\ f(b+1)=c+1 \ \wedge\ a-1=b$
  Note that now E_0 can be unsatisfiable.

- $$\begin{array}{rcl} a+2 & = & b-3 \\ b-5 & = & c+7 \\ c & = & d-4 \end{array} \qquad \text{is} \qquad \begin{array}{rcl} a & = & b-5 \\ b & = & c+12 \\ c & = & d-4 \end{array}$$

  An infinite number of classes, the ones of $\ldots, b-1, b, b+1, \ldots$
  can be represented by: $\{\ \mathbf{b} = a+5 = c+12 = d+8\}$

Integer Offsets (contd.)

- Can assume input equations of the form $a = b + k$ or of the form $\cdot(a, b + k_b) = c + k_c$ (not hard to see)
- Pending now contains eqs like $a = b+k$
- Representative(a) returns pair (b, k) such that $b = a+k$
- Similarly for Class lists, Lookup table, and Use lists.
- Obtain algorithm with same complexity!

BUT

If also atoms $s > t$ are allowed in (positive conjunction) input then satisfiability becomes NP-hard (reduce k-coloring, see paper for details).

CC-Ineq is NP-hard

Given a graph $G = (V, E)$, where $V = \{a_1, \ldots, a_n\}$ and $E = \{(b_1, b'_1), \ldots, (b_m, b'_m)\}$ and an integer k, the following CC-Ineq formula is satisfiable if and only if G is k-colorable:

$$G(c+1, c+1) = G(c+2, c+2) = \ldots = G(c+k, c+k) = true$$

$$\begin{array}{ccccc} c+k+1 & > & f(a_1) & > & c \\ c+k+1 & > & f(a_2) & > & c \\ \vdots & & \vdots & & \vdots \\ c+k+1 & > & f(a_n) & > & c \end{array} \qquad \begin{array}{ccc} true & > & G(f(b_1), f(b'_1)) \\ true & > & G(f(b_2), f(b'_2)) \\ \vdots & & \vdots \\ true & > & G(f(b_m), f(b'_m)) \end{array}$$

Intuitively, f represents the colour of each vertex (k possibilities), and G is used to express that no two adjacent vertices will have the same colour.

CC: our algorithm with offsets

While $Pending \neq \emptyset$ Do

  remove $a = b+k$ with representative $a' = b'+k_{b'}$ from $Pending$

  If $a' \neq b'$ and, wlog., $|ClassList(a')| \leq |ClassList(b')|$ Then

    For each $c+k_c$ in $ClassList(a')$ Do

      set $rep(c)$ to $(b', k_c - k_{b'})$ and add it to $ClassList(b')$

    EndFor

    For each $\cdot(c, d+k_d) = e+k_e$ in $UseList(a')$ Do

      If $Lookup(c', r(d+k_d))$ is $f+k_f$ and $r(f+k_f) \neq r(e+k_e)$ Then

        add $e = f+(k_f-k_e)$ to $Pending$

      EndIf

      set $Lookup(c', r(d+k_d)$ to $r(e+k_e)$

      add $\cdot(c, d+k_d) = e+k_e$ to $UseList(b')$

    EndFor

  ElseIf $a' = b'$ and $k_{b'} \neq 0$ Then return *unsatisfiable*

  EndIf

EndWhile