Congruence Closure and Extensions

Albert Oliveras

UPC, Barcelona

Joint work with Robert Nieuwenhuis

The University of Iowa

March 2005

Overview of this talk

1. Formulation of the problem
2. Union-Find
   - Abstract Union-Find
   - The Collapse rule
   - The Compose rule
   - Choosing a good ordering
3. Congruence closure (CC)
   - Abstract Congruence Closure
   - Initial transformations
   - General idea
   - Data structures and algorithm
   - Running example
   - Analysis of the algorithm
4. CC with integer offsets

Formulation of the problems

INPUT: set of ground equations E and $s{=}t$.

QUESTION: Is $E \models s{=}t$ true?

- Converting E into a convergent TRS will give us a decision procedure.
- Tiwari's Abstract Congruence Closure gives us a solution.
- Our goal: obtention of efficient strategies in practice.

Abstract Union-Find

- Signature $\Sigma = \{c_1, c_2, \ldots, c_n\}$ (only constants).
- Let $\succ$ a total ordering on constants, $c_1 \succ c_2 \succ \ldots \succ c_n$.

Orient $\dfrac{c = d, E}{c \to d, E}$ if $c \succ d$	Delete $\dfrac{c = c, E}{E}$
Simplify $\dfrac{c = d, c \to d', E}{d' = d, c \to d', E}$	
Collapse $\dfrac{c \to d', c \to d, E}{c \to d', d \to d', E}$ if $d \succ d'$	
Compose $\dfrac{c \to d, d \to d', E}{c \to d', d \to d', E}$	

- Any strategy will give us a convergent TRS, but, which is the most efficient one?

Abstract Union-Find(cntd.)

- Any strategy orienting all equations gives us a terminating TRS.
- Concerning confluence, which situations have to be avoided? (remember critical pair criterion)
- Our strategy: avoid situations where Collapse applies.
- Working with 4 rules instead of 5 is a reasonable way to make the implementation more efficient.

Avoiding applications of Collapse

- Which rules could transform a state in which Collapse does not apply into one in which it does?

Orient $\dfrac{c = d, E}{c \to d, E}$ if $c \succ d$	Delete $\dfrac{c = c, E}{E}$
Simplify $\dfrac{c = d, c \to d', E}{d' = d, c \to d', E}$	
Collapse $\dfrac{c \to d', c \to d, E}{c \to d', d \to d', E}$ if $d \succ d'$	
Compose $\dfrac{c \to d, d \to d', E}{c \to d', d \to d', E}$	

Only Orient and Compose (assuming we never apply Collapse).

Avoiding applications of Collapse (cntd.)

- Assume we move from a state in which Collapse doesn't apply to one in which it does. The new state has to be of the form $\{c \rightarrow d, c \rightarrow d', E\}$ with $d \succ d'$.
- Consider the case in which $c \rightarrow d'$ has just been generated (by Orient or Compose).
  - Case 1:

$$\text{Orient}\ \frac{c = d', \{c \rightarrow d, E'\}}{c \rightarrow d', \{c \rightarrow d, E'\}}$$

    but note that here Simplify also applies

$$\text{Simplify}\ \frac{c = d', c \rightarrow d, E'}{d = d', c \rightarrow d, E'}$$

    and now Collapse is not applicable.

    SOLUTION: Simplify has priority over Orient.

Avoiding applications of Collapse (cntd.)

- Assume we move from a state in which Collapse doesn't apply to one in which it does. The new state has to be of the form $\{c \to d, c \to d', E\}$ with $d > d'$.
- Consider the case in which $c \to d'$ has just been generated (by Orient or Compose).
  - Case 2:
  $$\text{Compose}\ \frac{c \to e, e \to d', \{c \to d, E'\}}{c \to d', e \to d', \{c \to d, E'\}}$$
  but note that e has to be d (why?) Thus, we have
  $$\text{Compose}\ \frac{c \to d, d \to d', E'}{c \to d', d \to d', E'}$$
  and in this new state Collapse does not apply.

Avoiding applications of Collapse (cntd.)

- Assume we move from a state in which Collapse doesn't apply to one in which it does. The new state has to be of the form $\{c \to d, c \to d', E\}$ with $d > d'$.
- The case in which $c \to d'$ has just been generated (by Orient or Compose) can be avoided by exhaustively applying Simplify before Orient.
- The case in which $c \to d$ has just been generated (by Orient or Compose) is similar.
- Using this strategy, we can get rid of Collapse, because it will never be applicable.

The Compose rule

- Remember: our goal is to obtain a convergent (confluent and terminating) TRS.
- Termination is ensured using any strategy.
- We have seen that if Simplify is exhaustively applied before Orient, all the intermediate states will give us confluent TRS.
- CONCLUSION: Compose is not necessary. So, let's forget about it (for the moment).

Choosing a good ordering

- According to the previous slides, our procedure will run the following loop until there is no unoriented equation:
  1. Pick an equation $c = d$.
  2. Apply Simplify exhaustively and get $c' = d'$.
  3. If c' is d', Delete. Otherwise, Orient it giving $c' \to d'$ if $c' > d'$.
- Complexity: $O(nL)$, being n is the number of equations and L the number of possible applications of Simplify to an equation.
- Instead of L, we can compute the maximal number of times a constant a can be rewritten. That is, the maximum length of a path of the form $a \to a_1 \to \ldots \to a_n$.
- **GOAL:** minimize the length of such a path

Choosing a good ordering (contd.)

- Given the ordering $a_1 \succ a_2 \succ \ldots \succ a_n$, and the equations $\{a_1 = a_2, a_2 = a_3, \ldots, a_{n-1} = a_n\}$ we can get a path of length n: $a_1 \rightarrow a_2 \rightarrow \ldots \rightarrow a_n$ (worst case!).
- Improvement: choose the ordering on the fly.
- Given a simplified equation $c = d$ (c and d are normal forms with respect to the TRS defined so far), its orientation will be $c \rightarrow d$ if $|c| \leq |d|$, being $|c|$ (resp. $|d|$) the number of constants in the equivalence class of c (resp. d).
- Since for each oriented rule $c \rightarrow d$, the class of $|c|$ at least doubled its size, any path has length at most $\lg n$, being n the number of constants.
- With these ordering restrictions, the complexity of the procedure is $O(m \lg n)$, being m the number of equations and n the number of constants.

Using Compose to improve the efficiency

- Imagine we pick the equation $c_0 = d$. We first have to Simplify it exhaustively: $c_0 \rightarrow c1 \rightarrow \ldots \rightarrow c_k$, being c_k its normal form.
- Later, we may pick another equation $c_0 = e$, and c_0 will have to be normalized using at least the previous k Simplify steps. Redundant work!!!
- **SOLUTION:** the first time we normalize c, we can at the same time apply Compose (compress the path) and get the oriented equations $c_i \rightarrow c_k$ for i in $0 \ldots k-1$.
- The next time we need to normalize c we will perform k steps in a single one, using the rule $c \rightarrow c_k$.
- This optimization, known as path-compression, allows one to run the procedure in time $O(m\alpha(m, n))$, where $\alpha(m, n)$ is a VERY slow-growing function.

Abstract Congruence Closure

INPUT: set of ground equations E and $s{=}t$.

QUESTION: Is $E \models s{=}t$ true?

- Equations in E and $s = t$ build over signature Σ consisting only of fixed-arity function symbols and constants.
- Rules are the ones of Abstract Union-Find plus:

Extend $\dfrac{s[f(c_1,\ldots,c_k)] = t, E}{s[c] = t, f(c_1,\ldots,c_k) \to c, E}$ if $f \in \Sigma$, $c \in K$	
Simplify $\dfrac{s[u] = t, u \to c, E}{s[c] = t, u \to c, E}$	
Superpose $\dfrac{f(c_1,\ldots,c_k) = c, f(c_1,\ldots,c_k) = d, E}{c = d, f(c_1,\ldots,c_k) = c, E}$	
Collapse $\dfrac{f(\ldots,c,\ldots) \to d, c \to c', E}{f(\ldots,c',\ldots) \to d, c \to c', E}$	Compose $\dfrac{f(\ldots) \to c, c \to d, E}{f(\ldots) \to d, c \to d, E}$

Initial transformations

First of all, two initial transformations are performed:

1. Curryfy (like in the implementation of FP):

$$\frac{s[f(c_1,\ldots,c_n)] = t, E}{s[\underbrace{\cdot(\cdot(\ldots\cdot(}_{n-1\ \ times}\cdot(f,c_1),c_2),\ldots,c_n)] = t, E}$$

- 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(Extend + Simplify):

- 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

$$\{\ \ \cdot(f,a) \to d,\ \ \cdot(g,b) \to e,\ \ \cdot(d,e) \to h,\ \ \cdot(h,c) = i\ \ \}$$

Reformulation of the problem

Now the CC problem is: $E \models a = b?$ $\quad (a, b, c, d, e$ cts.)

where equations in E are of the form $\cdot(c, d) = e$ or $c = d$.

The rules to be applied are the ones of the Abstract Union-Find plus:

Superpose $\dfrac{\cdot(c_1, c_2) \to c, \cdot(c_1, c_2) \to d, E}{c = d, \cdot(c_1, c_2) \to c, E}$	
Collapse$_1$ $\dfrac{\cdot(c_1, c_2) \to d, c_1 \to c_1', E}{\cdot(c_1', c_2) \to d, c_1 \to c_1', E}$	Collapse$_2$ $\dfrac{\cdot(c_1, c_2) \to d, c_2 \to c_2', E}{\cdot(c_1, c_2') \to d, c_2 \to c_2', E}$
Compose $\dfrac{\cdot(c_1, c_2) \to c, c \to d, E}{\cdot(c_1, c_2) \to d, c \to d, E}$	

Ideas behind the algorithm

- Due to the flattening process each term can now be identified with a constant. The question whether $s = t$ can be reduced to the question $c_s = c_t$ for certain constants c_s, c_t.
- Therefore, our goal is to detect which new equalities between constants arise due to the function symbols.
- In the rules, these new equalities are detected by Superpose.
- **IDEA**: we will need a Union-Find data structure and some procedure to detect these new equalities between constants.

Congruence closure: our data structures

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 $Pending \neq \emptyset$ 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

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:

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

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: $e_1 \to e_2$. We have $UseList(e_1) = \emptyset$.

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

$Constant$	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: $e_3 \rightarrow e_7$.

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

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

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

Running example (contd.)

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

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

$Constant$	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
$Representative$	a	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: $b \rightarrow c$.

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

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

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

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 $Pending$.

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

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

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

$\mathit{Constant}$	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
$\mathit{Representative}$	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: $e_2 \rightarrow e_7$.

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

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_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: $e_2 \to e_7$. Again $UseList(e_2) = \emptyset$.

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

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_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: $e_4 \to e_6$.

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

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: $e_4 \to e_6$. 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 $UseList(e_6)$.

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

$Constant$	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: $e_4 \to e_6$. Let's treat $\cdot(h, e_4) = e_5$. Since $\mathit{Lookup}(h', e'_4) = \mathit{Lookup}(h, e_6) = \emptyset$, just add $\mathit{Lookup}(h, e_6) = e_5$ to Lookup and $\cdot(h, e_4)$ to $\mathit{UseList}(e_6)$.

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

$\mathit{Constant}$	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
$\mathit{Representative}$	a	c	c	d	e_7	e_7	e_7	e_6	e_5	e_6	e_7	e_8	e_9

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

Running example (contd.)

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

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

$Constant$	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: $d \to e_6$ (and not the other way around).

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

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

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

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

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 $\mathit{Lookup}(h', d') = \mathit{Lookup}(h, e_6) = e_5$ and $e'_8 \neq e'_5$, add $e_5 = e_8$ to $\mathit{Pending}$. We also add $\cdot(h,d) = e_8$ to $\mathit{UseList}(e_6)$.

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

$\mathit{Constant}$	a	b	c	d	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
$\mathit{Representative}$	a	c	c	e_6	e_7	e_7	e_7	e_6	e_5	e_6	e_7	e_8	e_7

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

Running example (contd.)

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

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

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_5	e_6	e_7	e_8	e_7

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

Running example (contd.)

In normal form: $e_5 \to e_8$. Let's treat $\cdot(e_5, e_6) = e_7$. Since $\mathit{Lookup}(e_5', e_6') = \mathit{Lookup}(e_8, e_6) = e_9$, we should add $e_7' = e_9'$, but it is discarded because it already holds.

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

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

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

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.

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

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: $(RepresentativeE \cup Pending)^* = E_0^*$

Integer Offsets

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

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

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

- 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: our algorithm with offsets

```
While Pending ≠ ∅ Do
  remove a = b+k with representative a′ = b′+k_{b′} from Pending
  If a′ ≠ b′ and, wlog., |ClassList(a′)| ≤ |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 ·(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) ≠ 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 ·(c, d+k_d) = e+k_e to UseList(b′)
    EndFor
  ElseIf a′ = b′ and k_{b′} ≠ 0 Then return unsatisfiable
  EndIf
EndWhile
```

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.