Congruence Closure (CC)

An equational clause

$$\forall \vec{x}\,(t_1 \approx s_1 \vee \ldots \vee t_n \approx s_n \vee l_1 \not\approx r_1 \vee \ldots \vee l_k \not\approx r_k)$$

is valid iff

$$\exists \vec{x}\,(t_1 \not\approx s_1 \wedge \ldots \wedge t_n \not\approx s_n \wedge l_1 \approx r_1 \wedge \ldots \wedge l_k \approx r_k)$$

is unsatisfiable iff the Skolemized (ground!) formula

$$(t_1 \not\approx s_1 \wedge \ldots \wedge t_n \not\approx s_n \wedge l_1 \approx r_1 \wedge \ldots \wedge l_k \approx r_k)\{\vec{x} \mapsto \vec{c}\}$$

is unsatisfiable iff for the convergent TRS R out of $E = \{l_i \approx r_i \mid 1 \leq i \leq k\}$ there is an inequation $t_j \not\approx s_j$ such that $t_j \downarrow_R = s_j \downarrow_R$ or t_j and s_j are contained in the same congruence class modulo E. The first conidition can be checked by CC by rules, the latter by CC by terms.

Congruence Closure by Rules

The idea if the algorithm is to represent the ground E by a convergent TRS. For efficiency, common subterms are extracted, first. This is called *Flattening*. Let $E = l_1 \approx r_1 \wedge \ldots \wedge l_n \approx r_n$.

Flattening $E[f(t_1, \ldots, t_n)]_{p_1,\ldots,p_k} \;\Rightarrow_{\text{CCF}}$
$E[c/p_1, \ldots, p_k] \wedge f(t_1, \ldots, t_n) \approx c$
provided all t_i are constants, the p_j are all positions in E of $f(t_1, \ldots, t_n)$, $|p_k| > 2$ for some k, or, $p_k = n.2$ and $E|_{m.1}$ is not a constant for some n, and c is fresh

As a result: only two kinds of equations left.
Term equations: $f(c_{i_1}, \ldots, c_{i_n}) \approx c_{i_0}$
Constant equations: $c_i \approx c_j$.

The congruence closure algorithm is presented as a set of abstract rewrite rules operating on a pair of equations E and a set of rules R, $(E; R)$, similar to Knuth-Bendix completion, Section 4.4.

$(E_0; R_0) \Rightarrow_{\mathrm{CC}} (E_1; R_1) \Rightarrow_{\mathrm{CC}} (E_2; R_2) \Rightarrow_{\mathrm{CC}} \cdots$

At the beginning, $E = E_0$ is a set of constant equations and $R = R_0$ is the set of term equations oriented from left-to-right. At termination, E is empty and R contains the result.

Simplify $(E \uplus \{c \dot{\approx} c'\}; R \uplus \{c \rightarrow c''\}) \Rightarrow_{\text{CC}}$
$(E \cup \{c'' \dot{\approx} c'\}; R \cup \{c \rightarrow c''\})$

Delete $(E \uplus \{c \approx c\}; R) \Rightarrow_{\text{CC}} (E; R)$

Orient $(E \uplus \{c \dot{\approx} c'\}; R) \Rightarrow_{\text{CC}} (E; R \cup \{c \rightarrow c'\})$
if $c \succ c'$

Deduce $(E; R \uplus \{t \to c,\ t \to c'\}) \ \Rightarrow_{\text{CC}}$
$(E \cup \{c \approx c'\}; R \cup \{t \to c\})$

Collapse $(E; R \uplus \{t[c]_p \to c',\ c \to c''\}) \ \Rightarrow_{\text{CC}}$
$(E; R \cup \{t[c'']_p \to c',\ c \to c''\})$
$p \neq \epsilon$

For rule Deduce, t is either a term of the form $f(c_1, \ldots, c_n)$ or a constant c_i.
For rule Collapse, t is always of the form $f(c_1, \ldots, c_n)$

Congruence Closure by Terms

In contrast to congruence closure by rules that constructs a convergent TRS out of the ground equations, the traditional version of the Congruence Closure algorithm constructs an explicit representation of the equivalence classes.

The initial state is (Π, E), where Π is a partition of all ground terms, such that every term is in its own class, and E is the set of ground equations. The algorithm consists of the following three inference rules.

Delete $(\{A\} \cup \Pi, E \cup \{s \approx t\}) \Rightarrow_{CC} (\{A\} \cup \Pi, E)$
provided $\{s, t\} \subseteq A$.

Merge $(\{A, B\} \cup \Pi, E \cup \{s \approx t\}) \Rightarrow_{CC} (\{A \cup B\} \cup \Pi, E)$
provided $s \in A$, $t \in B$ and $A \neq B$.

Deduction $(\{A, B\} \cup \Pi, E) \Rightarrow_{CC}$
$(\{A, B\} \cup \Pi, E \cup \{f(s_1, ..., s_n) \approx f(t_1, ..., t_n)\})$
provided $f(s_1, ..., s_n) \in A$, $f(t_1, ..., t_n) \in B$, $A \neq B$ and for each i, there exists a $D_i \in \{A, B\} \cup \Pi$ such that $\{s_i, t_i\} \in D_i$ and $f(s_1, ..., s_n) \approx f(t_1, ..., t_n) \notin E$.

The algorithm terminates if no rule is applicable anymore. The resulting set Π represents the set of congruence classes. Assume for example the set $E = \{a \approx b, f(a) \approx g(a), f(b) \approx h(a)\}$. Initially the algorithm creates classes for each occuring term and subterm. Then Merge can be applied three times for the three equations. Since $a \approx b$ Deduct is applicable as well for $f(a)$ and $f(b)$. The final result is

$$(\{\{a, b\}, \{f(a), g(a), f(b), h(a), g(b), h(b)\}, \emptyset)$$