A decision procedure for equivalence relations

Sébastien Michelland
with Pierre Corbineau, Lionel Rieg and Karine Altisen

July 5, 2020

Congruence closure

$$\underbrace{f = g, \quad x = y, \quad f(x) = z}_{\text{Hypotheses}} \quad \vdash \quad \underbrace{g(y) = z}_{\text{Goal}}$$

- Terms: $\langle variable \rangle \mid \langle term \rangle \ \langle term \rangle$
- Deduction rules:

  Reflexivity
  Symmetry
  Transitivity

  $$\frac{f = g \quad x = y}{f(x) = g(y)} \ \textsc{Congruence}$$

- The congruence closure algorithm decides by saturation.

Congruence closure: example

$$f = g, \quad x = y, \quad f(x) = z \quad \vdash \quad g(y) = z$$

Congruence closure: example

$$f = g, \quad x = y, \quad f(x) = z \quad \vdash \quad g(y) = z$$

- Partition terms into equal class:

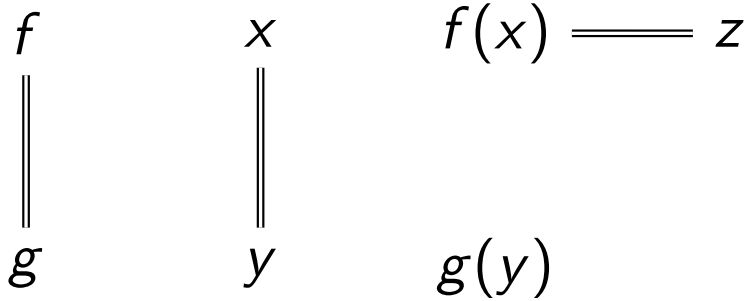
$$f \qquad x \qquad f(x) \qquad z$$

$$g \qquad y \qquad g(y)$$

Congruence closure: example

$$f = g, \quad x = y, \quad f(x) = z \quad \vdash \quad g(y) = z$$

- Partition terms into equal class:

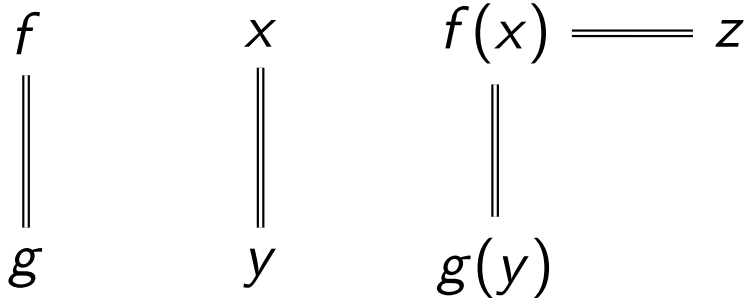


- The partition is reflexive, symmetric and transitive, so it saturates three out of four rules

Congruence closure: example

$$f = g, \quad x = y, \quad f(x) = z \quad \vdash \quad g(y) = z$$

- Partition terms into equal class:

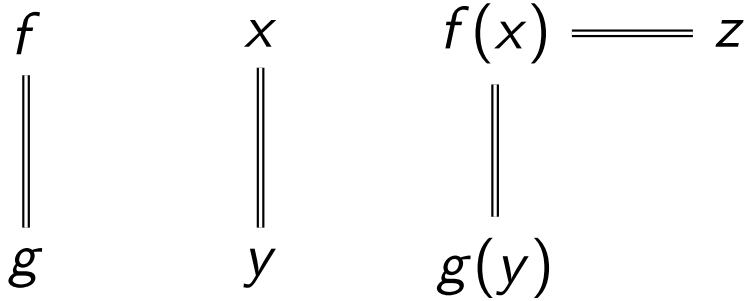


- The partition is reflexive, symmetric and transitive, so it saturates three out of four rules

Congruence closure: example

$$f = g, \quad x = y, \quad f(x) = z \quad \vdash \quad g(y) = z$$

- Partition terms into equal class:



- The partition is reflexive, symmetric and transitive, so it saturates three out of four rules
- No new terms are needed, the input is enough!
- Decides in quasi-linear time.

Proof generation

[NO05]

- "Deciding is cool but not as cool as proving" – Coq, probably

$$\begin{array}{ccc} f & x & f(x) = \text{Hyp} = z \\ \| & \| & \| \\ \text{Hyp} & \text{Hyp} & \text{CONGRUENCE} \\ \| & \| & \| \\ g & y & g(y) \end{array}$$

We can generate proof trees!

- $g(y) = z$ by transitivity, with $g(y) = f(x)$ by congruence (subproofs $g = f$ and $y = x$)

Limitations

[Cor06]

- congruence implemented in Coq by Pierre Corbineau (2001) (with extra features)

A couple limitations:

- In Coq $f = g$ is a definitional equality (not useful)
- Propositional equality $P = Q$ is also poor
- What about setoids and typeclasses?

Let's try equivalence relations.

Equivalence relations... and PERs

- Let's replace $=$ with equivalence relations.
- For functions, we'll use the respectful relation

$$f\ (R_1 \Rightarrow R_2)\ g\ \equiv\ \forall(x, y),\ x\ R_1\ y \rightarrow f(x)\ R_2\ g(y)$$

$$\frac{f\ (R_1 \Rightarrow R_2)\ g \qquad x\ R_1\ y}{f(x)\ R_2\ g(y)}\ \textsc{Congruence}$$

- But $R_1 \Rightarrow R_2$ is only symmetric and transitive, it's a partial equivalence relation (PER)!

Let's include PERs and improve the partition.

Completed relation of a PER

New idea: completed relation

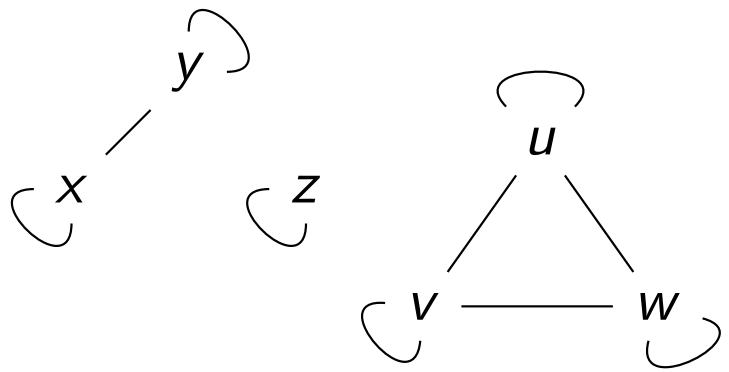
To represent a PER in a partition, consider

$$x\ \hat{R}\ y\ \equiv\ x\ R\ x \vee y\ R\ y \rightarrow x\ R\ y.$$

Isolated elements

Normal equivalence classes

$\boldsymbol{R}$: nr_1 nr_2 nr_3



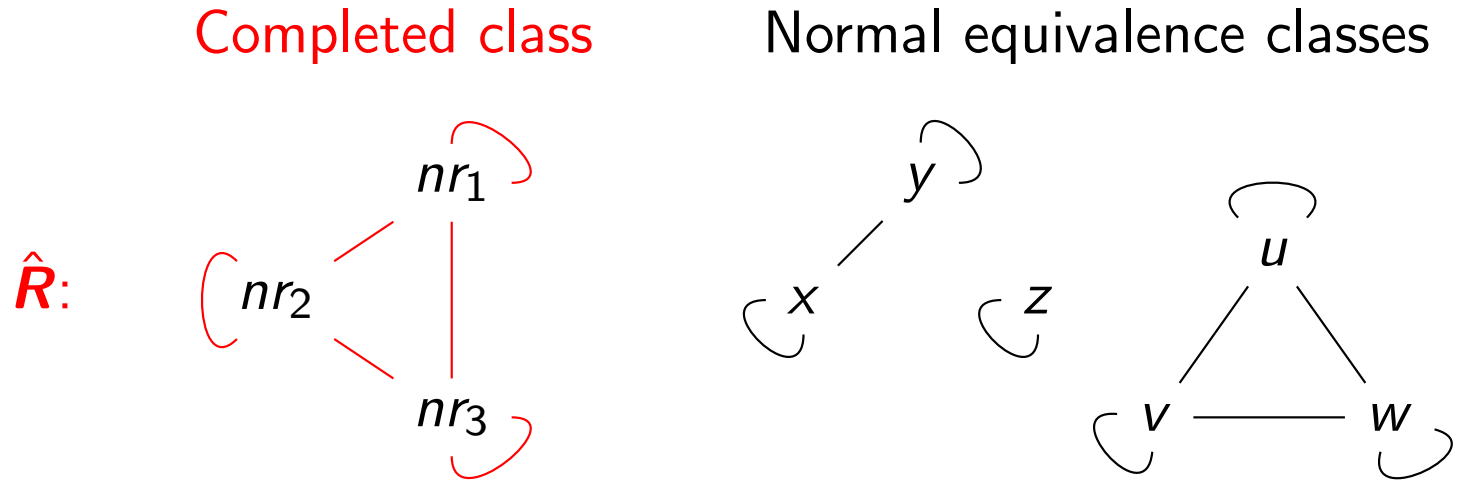
- $\hat{R}$ is a canonical equivalence relation associated with R

Completed relation of a PER

New idea: completed relation

To represent a PER in a partition, consider

$$x\ \hat{R}\ y\ \equiv\ x\ R\ x \vee y\ R\ y \rightarrow x\ R\ y.$$



- $\hat{R}$ is a canonical equivalence relation associated with R

Implementing the new congruence closure

- First version on the command-line
- Input: Coq-like text file, output: decision and proof tree
- Plus your everyday unit tests and coqc for proofs

We already gained some cool expressiveness!

Example:

- Equality of lists as multisets: $=_{\mathsf{MS}}$
- Concatenation preserves $=_{\mathsf{MS}}$: app $(=_{\mathsf{MS}} \Rightarrow =_{\mathsf{MS}} \Rightarrow =_{\mathsf{MS}})$ app

Relation inclusions

$$\text{List equality} \subseteq \text{Multiset equality} \subseteq \text{Set equality}$$

- Propagate each equality to larger relations

$$\frac{x\ R_1\ y \quad R_1 \subseteq R_2}{x\ R_2\ y}\ \textsc{Inclusion}$$

- Fast to saturate, so integrates nicely in the closure!

- PER compatibility : $R_1 \sqsubseteq R_2$ if operands of R_2 can be rewritten with equivalent terms for R_1.
- $R_1 \sqsubseteq R_2$ is equivalent to $R_1 \subseteq \hat{R}_2$, so we can saturate it!

Quantified hypotheses

$$\forall v_1 \dots v_n, x \ R \ y$$

- Very expressive!
  - Associativity: $\forall (l_1, l_2, l_3), l_1 + (l_2 + l_3) = (l_1 + l_2) + l_3$
  - Commutativity for $=_{\mathsf{MS}}$: $\forall (l_1, l_2), l_1 +\!\!+ \, l_2 =_{\mathsf{MS}} l_2 +\!\!+ \, l_1$

How to use them in the algorithm?

- Main concern: find $v_1 \dots v_n$ such that x or y is a known term.
- We then add $x \ R \ y$ and continue saturating (semi-decidable!)

Ematching to find $v_1 \dots v_n$

$$\forall x, f(e, x)\ R\ x \quad \longrightarrow \quad f\ e\ ?x\ \text{ and }\ ?x$$

- Find in a class C of R an instance of a pattern p:

  $$p \sim=_R C$$

  We want to find $v_1 \dots v_n$ and $t \in C$ such that $p(v_1 \dots v_n)\ R\ t$.

- By induction. For the inductive case

  $$f\ \langle pattern \rangle_1\ \dots\ \langle pattern \rangle_n,$$

  look only in classes that contain calls to f with n arguments (maintained like the signature table)

Combinatorial issues and termination

- Equalities are not oriented: termination issues

$$x = f(e, x) = f(e, f(e, x)) = \dots$$

- Risks of combinatorial explosion
  (Equivalence forms modulo associativity and commutativity!)

Hard questions and SMT heuristics!

- Here: we use ematching to find instances and stop after a fixed number of rounds
- Inefficient but just as expressive!

Conclusion

- Extended congruence closure for small-scale automation
- Equivalence relations, proofs, inclusions, even quantification
- Plugin is on the way, includes typeclasses
  $\rightsquigarrow$ Early testers welcome, contact us!

Future work:

- Use the setoid of propositions with `iff`
- Study relationships with SMT

Job offer!

- Post-doctoral position, 1 year
- Univ. Grenoble-Alpes, VERIMAG (France)
- Starting Fall 2020 (flexible)

Subject:

Coq formalization and proof techniques for distributed algorithms

- Contacts: Karine.Altisen@univ-grenoble-alpes.fr
Pierre.Corbineau@univ-grenoble-alpes.fr

Bibliography

Pierre Corbineau.
Deciding equality in the constructor theory.
In *International Workshop on Types for Proofs and Programs*, pages 78–92. Springer, 2006.

Robert Nieuwenhuis and Albert Oliveras.
Proof-producing congruence closure.
In *International Conference on Rewriting Techniques and Applications*, pages 453–468. Springer, 2005.

Coq session example

Demo.v

```
1  From Newcongruence Require Import Loader.
2
3  Require Import Morphisms.
4
5  (* Respectful elements to transport
   equalities through functions. *)
6  Lemma valid_congruence :
7    forall A B : Set,
8    forall x y : A,
9    forall RA : A -> A -> Prop,
10   forall RB : B -> B -> Prop,
11   Equivalence RA ->
12   Equivalence RB ->
13   forall f : A -> B,
14   (RA ==> RB)%signature f f ->
15   RA x y ->
16   RB (f(x)) (f(y)).
17 Proof.
18 intros.
19 setoid_congruence.
20 Abort.
21
```

```
1 subgoal
A, B : Set
x, y : A
RA : A -> A -> Prop
RB : B -> B -> Prop
H : Equivalence RA
H0 : Equivalence RB
f : A -> B
H1 : (RA ==> RB)%signature f f
H2 : RA x y
_____________________________________(1/1)
RB (f x) (f y)
```

Messages ↗ Errors ↗ Jobs ↗

```
Goal: (RB (f x) (f y))
Goal proved!
```