À la Nelson-Oppen Combination for congruence, LIA and LRA

F. Besson

Celtique/Inria/Univ Rennes

Juin 2021

On Automating Proofs

For limited fragments, decision procedures are available

- `congruence`: EUF + constructor
- `lia`: linear integer arithmetic
- `lra`: linear real arithmetic

No support for combination
EUF+LIA+LRA is *also* decidable ☹

Manual Combination

```
Goal ∀ (x y: Z) (P:Z -> Prop),
     x :: nil = y + 1 :: nil ->
     P (x - y) ->
     P 1.
Proof.
 intros.
 assert (x = y + 1) by congruence.
 assert (x - y = 1) by lia.
 congruence.
Qed.
```

Could we automate such proofs?

Yes! Instance of Nelson-Oppen combination scheme.

Contribution

A Nelson-Oppen combination scheme

- Parametrised *purification* engine
- Generic equality exchange engine
- Black-box call to existing tactics

A combination for EUF+LIA and EUF+LRA

Nelson-Oppen Scheme

1. Theories only share the equality symbol
2. Iterative exchange of equalities

There are conditions for completeness: convex, stably infinite.
EUF ✓ LRA ✓ LIA (non-convex)

How to Identify a Theory

A theory is identified by a signature.
⇒ We use type-classes to define symbols.

```
Class TheoryType(Tid:Type)(T:Type):Type.
Class TheorySig(Tid:Type){T:Type}(Op:T):Type.
```

- `Tid` is a theory identifier *e.g.*, `ZarithThy : Type.`
- `TheoryType` *e.g.*, `Z.`
- `TheorySig` *e.g.*, `Z.add.`

Purification

Definition (Pure Term)

A *pure* term belongs to a single theory.

Purification introduces fresh names and equations.

Exchange of equalities

Limitations

Conclusion