Isabelle's use of unification

When using `rule`, `erule`, `drule` Isabelle uses a process known as *higher-order unification*

Isabelle's use of unification

When using `rule`, `erule`, `drule` Isabelle uses a process known as *higher-order unification*

What is:

- Unification
- ...and specifically, higher-order unification?

Unification

Given two *terms* t, u defined over:

- Set of variables
- Constant symbols

can we find a *substitution* θ such that:

$$t\theta = u\theta$$

for some notion of equivalence or equality?

First-order unification

In *first-order unification* terms are *first-order terms*:

$$t, u, v ::= X \mid c \mid f(t_1, \ldots, t_n)$$

First-order unification

In *first-order unification* terms are *first-order terms*:

$$t, u, v ::= X \mid c \mid f(t_1, \ldots, t_n)$$

Equality is syntactic identity

Substitutions are finite functions from variables to terms

First-order unification

In *first-order unification* terms are *first-order terms*:

$$t, u, v ::= X \mid c \mid f(t_1, \ldots, t_n)$$

Equality is syntactic identity

Substitutions are finite functions from variables to terms

This process may be familiar:

- Part of operational semantics of logic programming (e.g. Prolog)
- Used widely in first-order theorem proving

Example

Suppose $+$ is a function symbol and 5 and 6 are constants

Suppose X and Y are variables

Example

Suppose $+$ is a function symbol and 5 and 6 are constants

Suppose X and Y are variables

Unify:

$$+(+(5,6),Y) \quad \text{with} \quad +(X,+(5,5))$$

Example

Suppose $+$ is a function symbol and 5 and 6 are constants

Suppose X and Y are variables

Unify:

$$+(+(5,6),Y) \quad \text{with} \quad +(X,+(5,5))$$

Solution $X \mapsto +(5,6)$ and $Y \mapsto +(5,5)$

Properties of first-order unification

Has many nice properties:

- Decidable
- Most general unifiers exist
- Linear-time algorithm via Martelli and Montanori

Isabelle's terms

Terms in Isabelle are typed λ-terms

Isabelle's terms

Terms in Isabelle are typed λ-terms

First-order unification inappropriate:

- Notion of equality is $\beta(\eta)$-equivalence
- Substitutions are *capture avoiding substitutions* from λ-calculus

Isabelle's terms

Terms in Isabelle are typed λ-terms

First-order unification inappropriate:

- Notion of equality is $\beta(\eta)$-equivalence
- Substitutions are *capture avoiding substitutions* from λ-calculus

Need higher-order unification...

Properties of higher-order unification

- Unifiability test is undecidable (Goldfarb and Huet)
- When unifiers do exist, most general unifiers need not
- Unifier set may be infinite

Properties of higher-order unification

- Unifiability test is undecidable (Goldfarb and Huet)
- When unifiers do exist, most general unifiers need not
- Unifier set may be infinite

Example:

Unify (where F is a variable of function type and c is a constant):

$$F\,c \quad \text{and} \quad c$$

Consider two different solutions:

$$F \mapsto \lambda x.c \quad \text{and} \quad F \mapsto \lambda x.x$$

Note $(\lambda x.c)c$ and $(\lambda x.x)c$ are both equivalent to c (in equational theory of simply-typed λ-calculus)

All is not lost!

Gerard Huet discovered a semi-decision procedure for higher-order unification in 1970s

Huet's algorithm:

- Finds unifiers when they exist
- May not terminate if unifiers do not exist
- Generally works well in practice

All is not lost!

Gerard Huet discovered a semi-decision procedure for higher-order unification in 1970s

Huet's algorithm:

- Finds unifiers when they exist
- May not terminate if unifiers do not exist
- Generally works well in practice

Most Isabelle unification problems are *pattern unification* problems:

- Decidable subfragment
- Discovered by Miller whilst working on λProlog
- Most general unifiers exist
- Efficient algorithms exist for pattern unification (Qian: linear time/space)

Isabelle uses pattern unification to reduce calls to Huet's algorithm