Extracting a formally verified Subtyping Algorithm for Intersection Types from Ideals and Filters

Jan Bessai

joint work with
Andrej Dudenhefner
Boris Düdder
Jakob Rehof

2016-05-23

technische universität
dortmund
fakultät für
informatik

Agenda

source: flickr/shawhargreaves

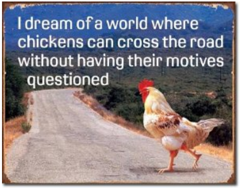
Deciding subtyping of intersection types in Coq:

- Why
- How
- For how long

.. did the chicken cross the intersection?
What is on the other side?

Motivation

- Machine verified and *assisted* reasoning for BCD Intersection Types (and Related Systems)
- Properties of subtyping dominant in most proofs
- Known subtyping algorithms [Hin82; Pie89; KT95; RU11; Sta15] are proven and implemented manually



source: amazon/posterrevolution

- Construction of (counter-)proofs instead of just yes/no
- Correct reference implementation for testing
- Meta logic elegance (pure Coq without axioms)
- Perhaps useful for others:
  - `https://github.com/JanBessai/BCD`

Intersection Types as an Inductive Type

```
Parameter 𝕍: Set
Parameter 𝕍_eq_dec:
  forall α β : 𝕍, { α = β } + { α <> β }.

Inductive IntersectionType : Set :=
| Var : 𝕍 -> IntersectionType
| Arr : IntersectionType -> IntersectionType -> IntersectionType
| Inter : IntersectionType -> IntersectionType -> IntersectionType
| Omega : IntersectionType.

Infix "→" := (Arr) (at level 88, right associativity).
Infix "∩" := (Inter) (at level 80, right associativity).
Definition ω := (Omega).
```

Encoding the Subtype Relation

```
Inductive SubtypeRules
  {R : IntersectionType -> IntersectionType -> Prop}:
    IntersectionType -> IntersectionType -> Prop :=
| R_InterMeetLeft : forall σ τ, σ ∩ τ ≤[R] σ
| R_InterMeetRight : forall σ τ, σ ∩ τ ≤[R] τ
| R_InterIdem : forall τ, τ ≤[R] τ ∩ τ
| R_InterDistrib : forall σ τ ρ,
  (σ → ρ) ∩ (σ → τ) ≤[R] σ → ρ ∩ τ
| R_SubtyDistrib: forall (σ σ' τ τ' : IntersectionType),
  R σ σ' -> R τ τ' -> σ ∩ τ ≤[R] σ' ∩ τ'
| R_CoContra : forall σ σ' τ τ',
  R σ σ' -> R τ τ' -> σ' → τ ≤[R] σ → τ'
| R_OmegaTop : forall σ, σ ≤[R] ω
| R_OmegaArrow : ω ≤[R] ω → ω
where "σ ≤[ R ] τ" := (SubtypeRules (R := R) σ τ).

Definition SubtypeRules_Closure
  {R : IntersectionType -> IntersectionType -> Prop}:
    IntersectionType -> IntersectionType -> Prop :=
  clos_refl_trans IntersectionType (@SubtypeRules R).
Notation "σ ≤*[ R ] τ" := (@SubtypeRules_Closure R σ τ) (at level 89).

Inductive Subtypes: IntersectionType -> IntersectionType -> Prop :=
| ST : forall σ τ, σ ≤* τ -> σ ≤ τ
where "σ ≤ τ" := (Subtypes σ τ)
and "σ ≤* τ" := (σ ≤*[Subtypes] τ).

Inductive EqualTypes : IntersectionType -> IntersectionType -> Prop :=
| InducedEq {σ τ}: σ ≤ τ -> τ ≤ σ -> σ ~= τ
where "σ ~= τ" := (EqualTypes σ τ).
```

Ω

A trick from [BCDC83] - inductively define Ω:

```
Inductive Ω: IntersectionType -> Prop :=
| OF_Omega : Ω ω
| OF_Arrow : forall σ ρ, Ω ρ -> Ω (σ → ρ)
| OF_Inter : forall σ ρ, Ω σ -> Ω ρ -> Ω (σ ∩ ρ)
where "↑ω σ" := (Ω σ).
```

And show that it is a directed upper set with ω as principal element, i.e. a principal filter:

```
Fact Ω_directed:
 forall σ τ, ↑ω σ -> ↑ω τ -> (↑ω ω) /\ (ω ≤ σ) /\ (ω ≤ τ).
Fact Ω_upperset:
 forall σ τ, σ ≤ τ -> ↑ω σ -> ↑ω τ.
Corollary Ω_principalElement:
 forall σ, ω ≤ σ -> ↑ω σ.
Fact Ω_principal:
 forall σ, ↑ω σ -> ω ~= σ.
```

We can easily prove

```
Lemma Beta_Omega: forall σ τ, ω ~= σ → τ <-> ω ~= τ.
Fact Ω_decidable: forall τ, { Ω τ } + { ~(Ω τ) }.
```

Predicate $\Omega(\tau)$ relies only on the syntax of τ

- No need to worry about transitive cuts ✌ ☺
- See [Lau12] for details on transitive cuts in $\leq$

Variable Ideals

We now follow this line of thought in a dual fashion and define principal ideals of variables:

```
Inductive VariableIdeal (α : 𝕍): IntersectionType -> Prop :=
| VI_Var : ↓α[α] (Var α)
| VI_InterLeft : forall σ τ, ↓α[α] σ -> ↓α[α] σ ∩ τ
| VI_InterRight : forall σ τ, ↓α[α] τ -> ↓α[α] σ ∩ τ
where "↓α[ α ] σ" := (VariableIdeal α σ).
```

This time we have:

```
Fact VariableIdeal_lowerset:
 forall σ τ, σ ≤ τ -> forall α, ↓α[α] τ -> ↓α[α] σ.
Corollary VariableIdeal_principalElement:
 forall σ α, σ ≤ (Var α) -> ↓α[α] σ.
Fact VariableIdeal_principal:
 forall α σ, ↓α[α] σ -> σ ≤ (Var α).
Fact VariableIdeal_directed:
 forall α σ τ,
   ↓α[α] σ -> ↓α[α] τ -> (↓α[α] (Var α)) /\ (σ ≤ (Var α)) /\ (τ ≤ (Var α)).
Fact VariableIdeal_prime:
  forall σ τ α, ↓α[α] σ ∩ τ -> (↓α[α] σ) \/ (↓α[α] τ).
```

Also easy now:

```
Lemma VariableIdeal_decidable: forall α τ, { ↓α[α] τ } + { ~(↓α[α] τ) }.
```

- Again: No cuts ✌ ☺

Arrow Ideals

```
Inductive ArrowIdeal (σ τ : IntersectionType): IntersectionType -> Prop :=
| AI_Omega : forall ρ, ↑ω τ -> ↓[σ] → [τ] ρ
| AI_Arrow : forall σ' τ', σ ≤ σ' -> τ' ≤ τ -> ↓[σ] → [τ] σ' → τ'
| AI_InterLeft : forall σ' τ', ↓[σ] → [τ] σ' -> ↓[σ] → [τ] σ' ∩ τ'
| AI_InterRight : forall σ' τ', ↓[σ] → [τ] τ' -> ↓[σ] → [τ] σ' ∩ τ'
| AI_Inter : forall σ' τ' ρ1 ρ2,
  ↓[σ] → [ρ1] σ' -> ↓[σ] → [ρ2] τ' -> ρ1 ∩ ρ2 ≤ τ -> ↓[σ] → [τ] σ' ∩ τ'
where "↓[ σ ] → [ τ ] ρ" := (ArrowIdeal σ τ ρ).
```

Again:

```
Fact ArrowIdeal_principal: forall σ τ ρ, ↓[σ] → [τ] ρ -> ρ ≤ σ → τ.
Fact ArrowIdeal_lowerset:
  forall ρ1 ρ2, ρ1 ≤ ρ2 -> forall σ τ, ↓[σ] → [τ] ρ2 -> ↓[σ] → [τ] ρ1.
Corollary ArrowIdeal_principalElement: forall ρ σ τ, ρ ≤ σ → τ -> ↓[σ] → [τ] ρ.
Fact ArrowIdeal_directed:
  forall ρ1 ρ2 σ τ, ↓[σ] → [τ] ρ1 -> ↓[σ] → [τ] ρ2 ->
    (↓[σ] → [τ] σ → τ) /\ (ρ1 ≤ σ → τ) /\ (ρ2 ≤ σ → τ).
```

No primality primality (yet), next best thing for now:

```
Fact ArrowIdeal_prime:
  forall ρ1 ρ2 σ τ,
  ↓[σ] → [τ] ρ1 ∩ ρ2 ->
  (((↓[σ] → [τ] ρ1) \/ (ρ2 ≤ ρ1)) \/ ((↓[σ] → [τ] ρ2) \/ (ρ1 ≤ ρ2)) <->
  (↓[σ] → [τ] ρ1) \/ (↓[σ] → [τ] ρ2)).
```

Decidability is also not immediate.

- Harder to deal with, because `AI_Arrow` and `AI_Inter` introduce cut types 😕

Full Ideals and Filters

```
Fixpoint Ideal σ: IntersectionType -> Prop :=
match σ with
| Omega => fun _ => Ω ω
| Var α => fun τ => ↓α[α] τ
| σ' → τ' => fun τ => ↓[σ'] → [τ'] τ
| σ' ∩ τ' => fun τ => (↓[σ'] τ) /\ (↓[τ'] τ)
end
where "↓[ σ ] τ" := (Ideal σ τ).
```

Again we get:

```
Lemma Ideal_principal: forall σ τ, ↓[σ] τ -> τ ≤ σ.
Lemma Ideal_lowerset:
  forall ρ1 ρ2, ρ1 ≤ ρ2 -> forall σ, ↓[σ] ρ2 -> ↓[σ] ρ1.
Lemma Ideal_principalElement:
  forall σ τ, τ ≤ σ -> ↓[σ] τ.
Lemma Ideal_directed:
  forall σ τ ρ, ↓[σ] τ -> ↓[σ] ρ -> (↓[σ] σ) /\ (τ ≤ σ) /\ (ρ ≤ σ).
```

And for free by dualization:

```
Definition Filter σ: IntersectionType -> Prop :=
match σ with
| Omega => Ω
| _ => fun τ => ↓[τ] σ
end.
Notation "↑[ σ ] τ" := (Filter σ τ) (at level 89).
Lemma Filter_Ideal: forall σ τ, ↑[σ] τ -> ↓[τ] σ.
Lemma Ideal_Filter: forall σ τ, ↓[σ] τ -> ↑[τ] σ.
```

Decision Procedure

Decreasing type sizes will be used to ensure termination:

```
Fixpoint ty_size σ : nat :=
match σ with
| Var _ => 1
| σ' → τ => ty_size σ' + ty_size τ
| ρ1 ∩ ρ2 => ty_size ρ1 + ty_size ρ2
| ω => 1
end.
Definition ty_pair_size στ : nat :=
  ty_size (fst στ) + ty_size (snd στ).
```

To deal with arrows we prove:

```
Fact Pick_Ideal: forall σ ρ
  (decσ : forall σ',
          ty_pair_size (σ, σ') < ty_pair_size (σ, ρ) ->
          { ↑[σ] σ' } + { ~(↑[σ] σ') } ),
  { τ : IntersectionType | (↓[σ] → [τ] ρ) /\
                           (forall τ', ↓[σ] → [τ'] ρ -> τ ≤ τ') /\
                           ty_size τ <= ty_size ρ }.
```

by induction on ρ, using

- τ' for $\rho \equiv \sigma' \to \tau'$ if $\uparrow [\sigma]\sigma'$
- $\tau_1 \cap \tau_2$ if $\rho \equiv \rho_1 \cap \rho_2$ for induction hypotheses yielding τ_1 and τ_2
- ω otherwise

Decision Procedure

Using the above we get:

```
Fact Pick_Ideal: forall σ ρ
  (decσ : forall σ',
    ty_pair_size (σ, σ') < ty_pair_size (σ, ρ) ->
      { ↑[σ] σ' } + { ~(↑[σ] σ') } ),
  { τ : IntersectionType | (↓[σ] → [τ] ρ) /\
                           (forall τ', ↓[σ] → [τ'] ρ -> τ ≤ τ') /\
                           ty_size τ <= ty_size ρ }.
Fact Ideal_decidable':
  forall στ
    (Ideal_decidable'':
      forall σ'τ',
        (ty_pair_size σ'τ' < ty_pair_size στ) ->
        { ↓[fst σ'τ'] (snd σ'τ') } + { ~(↓[fst σ'τ'] (snd σ'τ')) }),
    { ↓[fst στ] (snd στ) } + { ~(↓[fst στ] (snd στ)) }.
Lemma Ideal_decidable: forall σ τ, { ↓[σ] τ } + { ~(↓[σ] τ) }.
```

Where `Ideal_decidable'` uses:

- `Ω ω` if $\sigma \equiv \omega$
- `VariableIdeal_decidable` if $\sigma \equiv \alpha$
- Its recursive argument if $\sigma \equiv \sigma_1 \cap \sigma_2$
- Its recursive argument on the result of `Pick_Ideal` with `decσ` obtained by dualization of the recursive argument if $\sigma \equiv \sigma' \to \tau'$

`Ideal_decidable` is just the least fixedpoint over `Ideal_decidable'` ✌ ☺

Demo

Run Time

Additional (pre-)simplification of types possible:

- $((\alpha \cap \omega) \cap (\tau \to \omega)) \cap (\alpha \to \beta) = \alpha \cap (\alpha \to \beta)$

Proven by Andrej: if list concatenation in $\mathcal{O}(1)$ is available to us, we can get the algorithm down to $\mathcal{O}(n^2)$.

We can trigger the most expensive calls to `Pick_Ideal` with problems like

$$\bigcap_{i=1}^{k} (\sigma_i \to \sigma_i) \leq \bigcap_{i=1}^{k} (\bigcap_{\substack{j=1 \\ j \neq i}}^{k} \sigma_j \to \bigcap_{\substack{j=1 \\ j \neq i}}^{k} \sigma_j)$$

For now in the Coq version this (experimentally) leads to n^4 behavior if intersections are associated unfavorably.

Paths and Primality

Using the well-established concept of a path [RU11], we can show more about primality:

```
Inductive Path : IntersectionType -> Prop :=
| Path_Var : forall α, Path (Var α)
| Path_Arr : forall σ τ, Path τ -> Path (σ → τ).

Lemma Ideal_prime: forall τ,
  (forall ρ1 ρ2, ↓[τ] (ρ1 ∩ ρ2) -> (ρ1 ≤ τ) \/ (ρ2 ≤ τ)) <->
  exists τ', (τ ~= τ') /\ ((τ' ~= ω) \/ Path τ').
```

This turns organization into prime factoring [Sta15]:

```
Inductive Organized : IntersectionType -> Prop :=
| Organized_Path : forall τ, Path τ -> Organized τ
| Organized_Inter : forall σ τ, Path σ -> Organized τ -> Organized (σ ∩ τ).

Definition organization_lemma:
  forall τ, (τ ~= ω) + ({ τ': _ | Organized τ' /\ (τ ~= τ') }).
```

In contrast to natural numbers, idempotency of intersections will drop repeated prime factors. Tantalizing question:

- What about non idempotent intersection types?

Noise Reduction

These implementation techniques were helpful for noise reduction in proofs:

- Smart constructors:

```
Definition liftSubtypeProof {σ τ} (p : σ ≤[Subtypes] τ): σ ≤ τ :=
ST _ _ (rt_step _ _ _ _ p).

Definition InterMeetLeft {σ τ}: σ ∩ τ ≤ σ :=
  liftSubtypeProof (R_InterMeetLeft σ τ).
```

- Hint databases:

```
Create HintDb SubtypeHints.
Hint Resolve InterMeetLeft.

Example hints: forall α σ τ, (Var α) ∩ σ ∩ τ ≤ (Var α).
Proof.
  intros; auto with SubtypeHints.
Qed.
```

- Type classes and rewriting

```
Instance Subtypes_Reflexive : Reflexive (≤) := [...]
Instance Subtypes_Transitive : Transitive (≤) := [...]
Instance Inter_Proper_ST : Proper ((≤) ==> (≤) ==> (≤)) (∩) := [...]

Example rewriting: forall σ τ ρ, σ ≤ τ -> σ ∩ ρ ≤ τ ∩ ρ.
Proof.
  intros σ τ ρ σLEτ.
  rewrite σLEτ.
  reflexivity.
Qed.
```

Thanks ✌ ☺

References I

H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. "A Filter Lambda Model and the Completeness of Type Assignment". In: *Journal of Symbolic Logic* 48.4 (1983), pp. 931–940. DOI: 10.2307/2273659.

J. R. Hindley. "The Simple Semantics for Coppo-Dezani-Sallé Types". In: *International Symposium on Programming*. Vol. 137. LNCS. Springer, 1982, pp. 212–226.

T. Kurata and M. Takahashi. "Decidable properties of intersection type systems". In: *TLCA*. Vol. 902. LNCS. Springer, 1995, pp. 297–311.

References II

Olivier Laurent. "Intersection types with subtyping by means of cut elimination". In: *Fundamenta Informaticae* 121.1-4 (2012), pp. 203–226.

Benjamin C Pierce. *A decision procedure for the subtype relation on intersection types with bounded variables*. Tech. rep. CMU-CS-89-1693. CMU, 1989.

Jakob Rehof and Paweł Urzyczyn. "Finite Combinatory Logic with Intersection Types". In: *Proceedings of TLCA'11*. Vol. 6690. LNCS. Springer, 2011, pp. 169–183.

References III

Rick Statman. "A Finite Model Property for Intersection Types". In: *Proceedings Seventh Workshop on Intersection Types and Related Systems, ITRS 2014, Vienna, Austria, 18 July 2014.* 2015, pp. 1–9.