Determinism with Intersection and Union Types

Baber Rehman, Department of Computer Science, The University of Hong Kong
Supervisor: Dr. Bruno C. d. S. Oliveira

香 港 大 學
THE UNIVERSITY OF HONG KONG

Association for Computing Machinery

Motivation

Calculi with disjointness received significant research interest recently. The disjointness restricts potential problematic programs. An example is the disjoint intersection types proposed for deterministic merge operator. A variant of disjointness is also implemented in the type-based switch construct in Ceylon programming language. Such a type-based switch construct provides deterministic elimination of the union types. Another variant of disjointness has recently been implemented in Scala match types. The disjointness plays an integral role in maintaining the determinism in these calculi. We propose a novel disjointness algorithm for a calculus with intersection and union types. All of the metatheory has been formalized in Coq theorem prover.

Background

Intersection and union types are naturally able to encode various advance programming features such as function overloading, multiple interface inheritance, and nested composition. The merge operator $(e_1,,e_2)$ (Reynolds, 1997; Dunfield, 2014) is an introduction form for the intersection types. Unrestricted merge operator is not deterministic. Oliveira *et al.* (2016) study a restricted and deterministic form of the merge operator. The determinism is achieved by a notion of disjointness. We show their typing rule next:

$$\frac{\Gamma \vdash e_1 : A \qquad \Gamma \vdash e_2 : B \qquad A * B}{\Gamma \vdash e_1,,e_2 : A \wedge B}\ \text{TYP-MERG}$$

Oliveira *et al.* (2016)

Where $A * B$ indicates disjoint or non-overlapping types. For example, Int is disjoint with Bool. This study does not account for union types.

Surprisingly, a similar problem occurs in the elimination of (untagged) union types where multiple branches of a type-based switch construct overlap. The Ceylon programming language (King, 2013) implements a switch expression based on disjointness. Multiple branches of a switch expression cannot overlap in such an implementation. The disjointness in the presence of union types and type-based switch construct is formally studied by Rehman *et al.* (2022) in a calculus called λ_u:

$$\frac{\Gamma \vdash e : A \vee B \qquad \Gamma, x : A \vdash e_1 : C \qquad \Gamma, y : B \vdash e_2 : C \qquad A * B}{\Gamma \vdash \mathsf{switch}\ e\ \{(x : A) \to e_1, (y : B) \to e_2\} : C}\ \text{TYP-SWITCH}$$

Rehman *et al.* (2022)

Unfortunately, their extension with disjoint polymorphism results in a disjointness algorithm with ad-hoc restrictions on type variables. In particular, the type variable bounds are restricted to ground types:

$$\boxed{\boldsymbol{\Gamma, \alpha * G \vdash \ldots}}$$

$$\text{Ground Types } (G) \quad ::= \quad \top \mid \bot \mid \mathsf{Int} \mid A \to B \mid G_1 \vee G_2 \mid G_1 \wedge G_2 \mid \forall(\alpha * G).B$$

In summary, ground types lack type variables.

Such a restriction does not allow writing the following program, particularly `Y * X` is not allowed (`X` and `Y` are type variables):

```
//not allowed
firstMatch [X*Int, Y*X](x:X, y:Y){}
```

Novel Disjointness Algorithm

We study a novel disjointness algorithm for intersection and union types. This algorithm naturally extends for disjoint polymorphism without any ad-hoc restrictions!

$$\text{Types} \quad A, B, C \quad ::= \quad \top \mid \bot \mid \mathsf{Int} \mid A \to B \mid A \vee B \mid A \wedge B$$

$\boxed{A *_a B}$ *(Disjointness with intersection and union types)*

$$\frac{}{\bot *_a A}\ \text{AD-BTML} \qquad \frac{}{\mathsf{Int} *_a A \to B}\ \text{AD-INTL} \qquad \frac{A_1 \lhd A \rhd A_2 \qquad A_1 *_a B \qquad A_2 *_a B}{A *_a B}\ \text{AD-ORLL}$$

$$\frac{A_1 *_a B \qquad B^{\circledcirc}}{(A_1 \wedge A_2) *_a B}\ \text{AD-ANDLL} \qquad \frac{A_2 *_a B \qquad B^{\circledcirc}}{(A_1 \wedge A_2) *_a B}\ \text{AD-ANDLSS} \qquad \frac{A *_a B}{(A \wedge B) *_a C}\ \text{AD-EMPTYL}$$

Note that we do not show symmetric rules for simplicity.

$\boxed{B \lhd A \rhd C}$ *(Union splittable types)*

$$\frac{}{A \lhd A \vee B \rhd B}\ \text{USP-OR} \qquad \frac{A_1 \lhd A \rhd A_2}{A_1 \wedge B \lhd A \wedge B \rhd A_2 \wedge B}\ \text{USP-ORANDL} \qquad \frac{B_1 \lhd B \rhd B_2}{A \wedge B_1 \lhd A \wedge B \rhd A \wedge B_2}\ \text{USP-ORANDR}$$

$\boxed{A^{\circledcirc}}$ *(Union ordinary types)*

$$\frac{}{\bot^{\circledcirc}}\ \text{UO-BOT} \qquad \frac{}{\top^{\circledcirc}}\ \text{UO-TOP} \qquad \frac{}{\mathsf{Int}^{\circledcirc}}\ \text{UO-INT} \qquad \frac{}{(A \to B)^{\circledcirc}}\ \text{UO-ARROW} \qquad \frac{A^{\circledcirc} \qquad B^{\circledcirc}}{(A \wedge B)^{\circledcirc}}\ \text{UO-AND}$$

$\boxed{\Gamma \vdash A *_a B}$ *(Disjointness extension with polymorphism)*

$$\frac{\alpha * A \in \Gamma \qquad \Gamma \vdash B <: A}{\Gamma \vdash B *_a \alpha}\ \text{ADP-VARR} \qquad \frac{\alpha * A \in \Gamma \qquad \Gamma \vdash B <: A}{\Gamma \vdash \alpha *_a B}\ \text{ADP-VARL}$$

Our novel disjointness algorithm naturally extends for disjoint polymorphism and is more expressive. It accepts `firstMatch` function, i.e. it allows `Y * X`. We have also proved **type-safety** and **determinism** for such a calculus in Coq theorem prover. Interested readers may refer to Rehman *et al.* (2022) for the details on background.

Future Work

Future work includes the integration of two independent lines of research i.e. disjoint intersection types (Oliveira *et al.*, 2016) and disjoint switches (Rehman *et al.*, 2022). Such an integration turns out to be non-trivial and raises novel challenges in determinism.

References

Dunfield, Joshua. 2014. Elaborating intersection and union types. *Journal of Functional Programming*, **24**(2-3), 133–165.

King, Gavin. 2013. *The Ceylon language specification, version 1.0*.

Oliveira, Bruno C. d. S., Shi, Zhiyuan, & Alpuim, Joao. 2016. Disjoint intersection types. *Pages 364–377 of: Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*.

Rehman, Baber, Huang, Xuejing, Xie, Ningning, & Oliveira, Bruno C. d. S. 2022. Union Types with Disjoint Switches. *Pages 25:1–25:31 of: 36th European Conference on Object-Oriented Programming (ECOOP 2022)*, vol. 222.

Reynolds, John C. 1997. Design of the Programming Language F orsythe. *Pages 173–233 of: ALGOL-like languages*. Springer.