# Intersection Subtyping with Constructors

Olivier Laurent*
Univ Lyon, EnsL, UCBL, CNRS, LIP, F-69342, LYON Cedex 07, France
olivier.laurent@ens-lyon.fr

We study the question of extending the BCD intersection type system with additional type constructors. On the typing side, we focus on adding the usual rules for product types. On the subtyping side, we consider a generic way of defining a subtyping relation on families of types which include intersection types. We find back the BCD subtyping relation by considering the particular case where the type constructors are intersection, omega and arrow. We obtain an extension of BCD subtyping to product types as another instance. We show how the preservation of typing by both reduction and expansion is satisfied in all the considered cases. Our approach takes benefits from a "subformula property" of the proposed presentation of the subtyping relation.

## 1 Introduction

Intersection type systems are tools for building and analysing models of the $\lambda$-calculus [BCDC83, Bak95, RDRP04, ABDC06]. They also provide ways of characterising reduction properties of $\lambda$-terms such as normalization. The main difference to other type systems is the fact that not only *subject reduction* holds (if $t$ reduces to $u$ and $\Gamma \vdash t : A$ then $\Gamma \vdash u : A$) but also *subject expansion* holds (if $t$ reduces to $u$ and $\Gamma \vdash u : A$ then $\Gamma \vdash t : A$). As a consequence it is possible to define a denotational model by associating to each (closed) term the set of its types $[\![t]\!] = \{A \mid \vdash t : A\}$.

The most famous intersection type system is probably the BCD system [BCDC83], and this is the one we are focusing on. While BCD insists on the interaction between arrow types and intersection types, we want to consider more general sets of type constructors, following [BCD$^+$18]. The BCD type system can be decomposed into two parts: typing rules and subtyping rules. They are related through the subsumption rule. Our main contribution is a derivation system for the subtyping relation which allows us to deal with generic type constructors while satisfying a "subformula property". Alternative presentations of BCD subtyping are studied in [Ven94]. In contrast with [BCD$^+$18], we allow contravariant type constructors so that even the arrow constructor can be defined as an instance of our generic pattern, and only intersection has a specific status.

In Section 2, we recall standard syntactic proofs [ABDC06, Lau12] of preservation of typing by $\beta$-reduction and $\beta$-expansion for the BCD system. Our presentation stresses the fact that, starting from intersection (and $\Omega$) only, type constructors can be added in a modular way. In Section 2.2, we consider the arrow types, thus obtaining the usual BCD rules. We extend the results to product types in Section 2.3. The main part of the paper is then Section 3 where we propose a sequent-style derivation system for defining BCD-like subtyping relations for extensions of intersection types to generic sets of constructors. Starting from a transitivity/cut admissibility property, we prove that instances of our system are equivalent with variants of the BCD subtyping relation.

*This work was supported by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007), and by the project Elica (ANR-14-CE25-0005), both operated by the French National Research Agency (ANR). This work was also supported by GDRI Linear Logic.

S. Alves and M. Pagani (Eds.): DCM 2018 and ITRS 2018
EPTCS 293, 2019, pp. 73–84, doi:10.4204/EPTCS.293.6

$$\frac{}{\Gamma, x : A, \Delta \vdash x : A}\ var \qquad \frac{\Gamma \vdash t : A \qquad A \leq B}{\Gamma \vdash t : B} \leq \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash t : B}{\Gamma \vdash t : A \cap B} \cap \qquad \frac{}{\Gamma \vdash t : \Omega}\ \Omega$$

**Table 1:** Typing Rules with Subtyping and Intersection.

Key results on subtyping (Propositions 5 and 6, and Theorem 1) are formalized in `Coq`:

`https://perso.ens-lyon.fr/olivier.laurent/bcdc/`

# 2 Intersection Typing

We present the system we are looking at, which is mainly BCD [BCDC83] extended with product types. Type constructors are introduced in an incremental and modular way.

## 2.1 Intersection Types

Let us first consider an at most countable set $\mathscr{X}$ of base types denoted $X$, $Y$, etc, and consider types built using at least the following constructors:

$$A, B ::= X \mid A \cap B \mid \Omega \mid \dots$$

Similarly we do not define the exact set of terms (denoted $t$, $u$, etc), but first we only assume they contain a denumerable set of term variables $\mathscr{V}$ (whose elements are denoted $x$, $y$, etc). A first set of typing rules is given on Table 1, with judgements $\Gamma \vdash t : A$ built from a list $\Gamma$ of typing declarations for variables (of the shape $x : B$), a term $t$ and a type $A$. Note these rules rely on a *subtyping* relation $\leq$ on types, which is here just a parameter.

**Lemma 1** (Weakening)
*If $\Gamma \vdash t : A$ and $\Delta \leq \Gamma$ (meaning that, for each $x : B$ in $\Gamma$, one can find $x : B'$ in $\Delta$ with $B' \leq B$) then $\Delta \vdash t : A$.*

**Lemma 2** (Strengthening)
*If $\Gamma, x : B, \Delta \vdash t : A$ and $x \notin t$ then $\Gamma, \Delta \vdash t : A$.*

Because it makes hypotheses on the term in conclusion, the rule ($var$) is called a *term rule* (the introduced term must be a variable). In the opposite, ($\leq$), ($\cap$) and ($\Omega$) rules are called *non-term* as they apply on any term without any constraint on its main constructor. As a term rule, ($var$) admits a so-called *generation lemma* analysing how variables can by typed. For this, we make some hypotheses on the subtyping relation (see Table 2).

Note in passing, that the axioms of Table 2 are equivalent to say that $\leq$ is a preorder relation with $\cap$ as greatest lower bound and $\Omega$ as top element. In particular, up to the equivalence relation induced by $\leq$, $\cap$ is a commutative associative idempotent operation with $\Omega$ as unit. As a consequence the notation $\bigcap_{i \in I} A_i$ makes sense (up to the equivalence relation induced by $\leq$) for any (possibly empty) finite set $I$.

**Lemma 3** (Generation for Variables)
*Assuming that (var) is the only term rule introducing a variable, the only non-term rules are ($\leq$), ($\cap$) and ($\Omega$), and that the axioms of Table 2 are satisfied, we have: if $\Gamma \vdash x : A$ with $x : B \in \Gamma$ then $B \leq A$.*

**Lemma 4** (Substitution)
*If $\Gamma, x : A, \Delta \vdash t : B$ and $\Gamma, \Delta \vdash u : A$ then $\Gamma, \Delta \vdash t[^u/_x] : B$.*

| | |
|---|---|
| $A \leq A$ | (*refl*) |
| $A \leq B \wedge B \leq C \Rightarrow A \leq C$ | (*trans*) |
| $A \cap B \leq A$ | $(\cap_l^1)$ |
| $A \cap B \leq B$ | $(\cap_l^2)$ |
| $C \leq A \wedge C \leq B \Rightarrow C \leq A \cap B$ | $(\cap_r)$ |
| $A \leq \Omega$ | $(\Omega_r)$ |

**Table 2:** Kernel Properties of Subtyping.

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ abs \qquad\qquad \frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\,u : B}\ app$$

**Table 3:** Typing Rules for Arrow.

## 2.2 Arrow Types

We now assume types contain an arrow constructor and terms are extended correspondingly:

$$A, B ::= X \mid A \cap B \mid \Omega \mid A \to B \mid \dots \qquad\qquad t, u ::= x \mid \lambda x.t \mid t\,u \mid \dots$$

The associated typing rules are given on Table 3. Note the two new rules are term rules corresponding respectively to $\lambda x.t$ and $t\,u$ (no new non-term rule).

By adding new cases corresponding to the added rules in the proofs, one can check that Lemmas 1 and 2 still hold. Moreover the hypotheses of Lemma 3 are still satisfied, and finally Lemma 4 (which only relies on the previous lemmas) is still true as well. Since terms may now contain binders (such as $\lambda x.\_$), substitution has to be considered as being capture-avoiding substitution.

**Lemma 5** (Generation for Application)
*Assuming that (app) is the only term rule introducing an application, the only non-term rules are ($\leq$), ($\cap$) and ($\Omega$), and that the axioms of Table 2 are satisfied, we have: if $\Gamma \vdash t\,u : B$ then there exist two families of types $(A_i)_{i \in I}$ and $(B_i)_{i \in I}$ with $\bigcap_{i \in I} B_i \leq B$ and, for each $i \in I$, $\Gamma \vdash t : A_i \to B_i$ and $\Gamma \vdash u : A_i$.*

**Lemma 6** (Generation for Abstraction)
*Assuming that (abs) is the only term rule introducing an abstraction, the only non-term rules are ($\leq$), ($\cap$) and ($\Omega$), and that the axioms of Table 2 are satisfied, we have: if $\Gamma \vdash \lambda x.t : A$ then there exist two families of types $(B_i)_{i \in I}$ and $(C_i)_{i \in I}$ with $\bigcap_{i \in I}(B_i \to C_i) \leq A$ and, for each $i \in I$, $\Gamma, x : B_i \vdash t : C_i$.*

We now have the requested material to prove subject reduction and subject expansion. However a specific property on subtyping is still missing:

$$\bigcap_{i \in I}(A_i \to B_i) \leq A \to B \Rightarrow \exists J \subseteq I, \left(\bigcap_{i \in J} B_i \leq B \wedge \forall i \in J, A \leq A_i\right) \qquad (\to\leq\to)$$

The study of this property will be at the heart of Section 3.

$$\frac{}{A \leq A} \qquad \frac{A \leq B \qquad B \leq C}{A \leq C} \qquad \frac{}{A \leq \Omega}$$

$$\frac{}{A \cap B \leq A} \qquad \frac{}{A \cap B \leq B} \qquad \frac{}{A \leq A \cap A} \qquad \frac{A \leq C \qquad B \leq D}{A \cap B \leq C \cap D}$$

$$\frac{C \leq A \qquad B \leq D}{A \to B \leq C \to D} \qquad \frac{}{(A \to B) \cap (A \to C) \leq A \to (B \cap C)} \qquad \frac{}{\Omega \leq \Omega \to \Omega}$$

**Table 4:** BCD Subtyping Rules.

**Proposition 1** (Subject Reduction)
*Assuming* $(\to\leq\to)$*, if* $t_1 \to_\beta t_2$ *and* $\Gamma \vdash t_1 : A$ *then* $\Gamma \vdash t_2 : A$*.*

*Proof.* The key case is $(\lambda x.t)\,u \to_\beta t[^u/_x]$. If $\Gamma \vdash (\lambda x.t)\,u : A$, by Lemma 5, we have two families $(B_i)_{i\in I}$ and $(C_i)_{i\in I}$ with $\bigcap_{i\in I} C_i \leq A$ and, for each $i \in I$, $\Gamma \vdash \lambda x.t : B_i \to C_i$ and $\Gamma \vdash u : B_i$. For each $i \in I$, by Lemma 6, we have two families $(B'_j)_{j\in J_i}$ and $(C'_j)_{j\in J_i}$ with $\bigcap_{j\in J_i}(B'_j \to C'_j) \leq B_i \to C_i$ and, for each $j \in J_i$, $\Gamma, x : B'_j \vdash t : C'_j$. By $(\to\leq\to)$, there exists $K_i \subseteq J_i$ such that $B_i \leq B'_j$ ($j \in K_i$) and $\bigcap_{j\in K_i} C'_j \leq C_i$. We conclude by using Lemma 4 with $\Gamma \vdash u : B'_j$:

$$\dfrac{\dfrac{\cdots \qquad \dfrac{\dfrac{\cdots \quad \Gamma \vdash t[^u/_x] : C'_j \quad \cdots}{\Gamma \vdash t[^u/_x] : \bigcap_{j\in K_i} C'_j}\,\cap \qquad \bigcap_{j\in K_i} C'_j \leq C_i}{\Gamma \vdash t[^u/_x] : C_i}\,\leq \qquad \cdots}{\Gamma \vdash t[^u/_x] : \bigcap_{i\in I} C_i}\,\cap \qquad \bigcap_{i\in I} C_i \leq A}{\Gamma \vdash t[^u/_x] : A}\,\leq$$

□

**Proposition 2** (Subject Expansion)
*If* $t_1 \to_\beta t_2$ *and* $\Gamma \vdash t_2 : A$ *then* $\Gamma \vdash t_1 : A$*.*

*Proof.* The key case is $(\lambda x.t)\,u \to_\beta t[^u/_x]$. We first prove that $\Gamma \vdash t[^u/_x] : B$ implies that we can find a type $A$ such that $\Gamma, x : A \vdash t : B$ and $\Gamma \vdash u : A$, by induction on the derivation of $\Gamma \vdash t[^u/_x] : B$. And then:

$$\dfrac{\dfrac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\,abs \qquad \Gamma \vdash u : A}{\Gamma \vdash (\lambda x.t)\,u : B}\,app$$

□

To sum up, we have shown that, given the typing rules of Tables 1 and 3, the subject reduction and subject expansion properties hold for $\beta$-reduction as soon as the chosen subtyping satisfies the axioms of Table 2 as well as property $(\to\leq\to)$. The historical example from the literature is the BCD system [BCDC83] corresponding to the subtyping relation of Table 4. We will come back to the fact that $(\to\leq\to)$ holds for this BCD relation (Lemma 11).

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash \langle t, u \rangle : A \times B}\ pair \qquad \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \pi_1 t : A}\ proj_1 \qquad \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \pi_2 t : B}\ proj_2$$

**Table 5:** Typing Rules for Product.

## 2.3 Product Types

We now assume types contain a product constructor and terms are extended correspondingly:

$$A, B ::= X \mid A \cap B \mid \Omega \mid A \to B \mid A \times B \mid \dots \qquad t, u ::= x \mid \lambda x.t \mid t\,u \mid \langle t, u \rangle \mid \pi_1 t \mid \pi_2 t \mid \dots$$

The associated typing rules are given on Table 5. Note the new rules are all term rules (no non-term rule added). Lemmas 1, 2, 3 and 4 still hold. It is also easy to check that the new rules do not break Propositions 1 and 2.

**Lemma 7** (Generation for Pairing)
*Assuming that (pair) is the only term rule introducing a pair, the only non-term rules are ($\leq$), ($\cap$) and ($\Omega$), and that the axioms of Table 2 are satisfied, we have: if $\Gamma \vdash \langle t, u \rangle : A$ then there exist two families of types $(B_i)_{i \in I}$ and $(C_i)_{i \in I}$ with $\bigcap_{i \in I}(B_i \times C_i) \leq A$ and, for each $i \in I$, $\Gamma \vdash t : B_i$ and $\Gamma \vdash u : C_i$.*

*Proof.* By induction on the typing derivation of $\Gamma \vdash \langle t, u \rangle : A$, by looking at each possible last rule which, by assumption, can only be (*pair*), ($\leq$), ($\cap$) or ($\Omega$):

- (*pair*) rule: $I$ is a singleton and the result is immediate.
- ($\leq$) rule: we have $\Gamma \vdash \langle t, u \rangle : A'$ with $A' \leq A$, we apply the induction hypothesis to $\Gamma \vdash \langle t, u \rangle : A'$, and we conclude by (*trans*).
- ($\cap$) rule: we have $A = A' \cap A''$, by induction hypotheses we obtain families of types indexed by $I'$ and $I''$ and we consider $I = I' \uplus I''$. We conclude by using $E_1 \leq E_2 \wedge F_1 \leq F_2 \Rightarrow E_1 \cap F_1 \leq E_2 \cap F_2$.
- ($\Omega$) rule: we simply choose $I = \emptyset$. □

**Lemma 8** (Generation for Left Projection)
*Assuming that ($proj_1$) is the only term rule introducing a left projection, the only non-term rules are ($\leq$), ($\cap$) and ($\Omega$), and that the axioms of Table 2 are satisfied, we have: if $\Gamma \vdash \pi_1 t : A$ then there exist two families of types $(B_i)_{i \in I}$ and $(C_i)_{i \in I}$ with $\bigcap_{i \in I} B_i \leq A$ and, for each $i \in I$, $\Gamma \vdash t : B_i \times C_i$.*

*Proof.* By induction on the typing derivation of $\Gamma \vdash \pi_1 t : A$, by looking at each possible last rule which, by assumption, can only be ($proj_1$), ($\leq$), ($\cap$) or ($\Omega$):

- ($proj_1$) rule: $I$ is a singleton and the result is immediate.
- ($\leq$) rule: we have $\Gamma \vdash \pi_1 t : A'$ with $A' \leq A$, we apply the induction hypothesis to $\Gamma \vdash \pi_1 t : A'$, and we conclude by (*trans*).
- ($\cap$) rule: we have $A = A' \cap A''$, by induction hypotheses we obtain families of types indexed by $I'$ and $I''$ and we consider $I = I' \uplus I''$. We conclude by using $E_1 \leq E_2 \wedge F_1 \leq F_2 \Rightarrow E_1 \cap F_1 \leq E_2 \cap F_2$.
- ($\Omega$) rule: we simply choose $I = \emptyset$. □

**Lemma 9** (Generation for Right Projection)
*Assuming that (proj$_2$) is the only term rule introducing a right projection, the only non-term rules are ($\leq$), ($\cap$) and ($\Omega$), and that the axioms of Table 2 are satisfied, we have: if $\Gamma \vdash \pi_2 t : A$ then there exist two families of types $(B_i)_{i\in I}$ and $(C_i)_{i\in I}$ with $\bigcap_{i\in I} C_i \leq A$ and, for each $i \in I$, $\Gamma \vdash t : B_i \times C_i$.*

*Proof.* Similar to the proof of Lemma 8. □

We consider the reduction $\to_\pi$ to be the congruence generated by:

$$\pi_1 \langle t,u\rangle \to_\pi t \qquad\qquad \pi_2 \langle t,u\rangle \to_\pi u$$

Similarly to the arrow case, we ask for an additional property of the subtyping relation in order to deduce subject reduction:

$$\bigcap_{i\in I}(A_i \times B_i) \leq A \times B \Rightarrow \bigcap_{i\in I} A_i \leq A \wedge \bigcap_{i\in I} B_i \leq B \qquad (\times \leq \times)$$

**Proposition 3** (Subject Reduction for Products)
*Assuming $(\times \leq \times)$, if $t_1 \to_\pi t_2$ and $\Gamma \vdash t_1 : A$ then $\Gamma \vdash t_2 : A$.*

*Proof.* The key case is $\pi_1 \langle t,u\rangle \to_\pi t$. If $\Gamma \vdash \pi_1 \langle t,u\rangle : A$, by Lemma 8, we have two families $(B_i)_{i\in I}$ and $(C_i)_{i\in I}$ with $\bigcap_{i\in I} B_i \leq A$ and, for each $i \in I$, $\Gamma \vdash \langle t,u\rangle : B_i \times C_i$. For each $i \in I$, by Lemma 7, we have two families $(B'_j)_{j\in J_i}$ and $(C'_j)_{j\in J_i}$ with $\bigcap_{j\in J_i}(B'_j \times C'_j) \leq B_i \times C_i$ and, for each $j \in J_i$, $\Gamma \vdash t : B'_j$ and $\Gamma \vdash u : C'_j$. By $(\times \leq \times)$, $\bigcap_{j\in J_i} B'_j \leq B_i$. We conclude by:

$$\dfrac{\cdots \qquad \dfrac{\dfrac{\dfrac{\cdots \quad \Gamma \vdash t : B'_j \quad \cdots}{\Gamma \vdash t : \bigcap_{j\in J_i} B'_j}\cap \qquad \bigcap_{j\in J_i} B'_j \leq B_i}{\Gamma \vdash t : B_i}\leq \qquad \cdots}{\Gamma \vdash t : \bigcap_{i\in I} B_i}\cap \qquad \bigcap_{i\in I} B_i \leq A}{\Gamma \vdash t : A}\leq$$

□

**Proposition 4** (Subject Expansion for Products)
*If $t_1 \to_\pi t_2$ and $\Gamma \vdash t_2 : A$ then $\Gamma \vdash t_1 : A$.*

*Proof.* The key case is $\pi_1 \langle t,u\rangle \to_\pi t$. We have:

$$\dfrac{\dfrac{\Gamma \vdash t : A \qquad \dfrac{}{\Gamma \vdash u : \Omega}\,\Omega}{\Gamma \vdash \langle t,u\rangle : A \times \Omega}\,pair}{\Gamma \vdash \pi_1 \langle t,u\rangle : A}\,proj_1$$

□

Following [BCD+18] in extending BCD subtyping in the context of additional type constructors, we can consider the rules of Table 6 for subtyping with products. This system satisfies property $(\times \leq \times)$ (Lemma 12).

While the present section focused on the product extension of BCD, our purpose is to use it as a concrete application of a more general pattern of subtyping between types which include intersection as well as other type constructors. What should be remembered from what we have done so far, is that we can get subject reduction and subject expansion as soon as the subtyping relation satisfies Table 2 as well as $(\to\leq\to)$ and $(\times \leq \times)$. The next section provides a general approach to these results.

$$\frac{A \leq C \qquad B \leq D}{A \times B \leq C \times D} \qquad\qquad \frac{}{(A \times B) \cap (C \times D) \leq (A \cap C) \times (B \cap D)}$$

**Table 6:** BCD-Style Subtyping Rules for Products.

$$\frac{\Gamma, \Delta \vdash C}{\Gamma, \kappa(\vec{A};\vec{B}), \Delta \vdash C}\, wk \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \cap B}\, \cap R \qquad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \cap B, \Delta \vdash C}\, \cap L$$

$$\frac{\begin{array}{ccc} A_1 \vdash A_1^1 \quad \cdots \quad A_1 \vdash A_1^k & B_1^1, \dots, B_1^k \vdash B_1 & \\ \vdots & \vdots & \\ A_{\alpha_\kappa} \vdash A_{\alpha_\kappa}^1 \quad \cdots \quad A_{\alpha_\kappa} \vdash A_{\alpha_\kappa}^k & B_{\beta_\kappa}^1, \dots, B_{\beta_\kappa}^k \vdash B_{\beta_\kappa} & \varepsilon(\kappa) \leq k \end{array}}{\kappa(A_1^1, \dots, A_{\alpha_\kappa}^1; B_1^1, \dots, B_{\beta_\kappa}^1), \dots, \kappa(A_1^k, \dots, A_{\alpha_\kappa}^k; B_1^k, \dots, B_{\beta_\kappa}^k) \vdash \kappa(A_1, \dots, A_{\alpha_\kappa}; B_1, \dots, B_{\beta_\kappa})}\, constr$$

**Table 7:** ISC Deduction System.

# 3 Intersection Subtyping

Inspired by [BCD$^+$18], we directly consider types built with an arbitrary set of constructors. The case of $\times$ for example will be obtained as a particular instance. We go in fact one step further than [BCD$^+$18] by allowing enough generality in the treatment of constructors so that $\rightarrow$ appears as a constructor among others and not as a specific one as given in [BCD$^+$18].

## 3.1 Generic Subtyping with Constructors

We consider a given set $\mathscr{K}$ of *type constructors* (denoted $\kappa$, $\kappa_1$, $\kappa_2$, etc) which come with a *contravariant arity* $\alpha_\kappa$ and a *covariant arity* $\beta_\kappa$. We assume that arities are respected when constructing types, so that if $\alpha_\kappa = 2$ and $\beta_\kappa = 1$, then $\kappa(A,B;C)$ is a type when $A$, $B$ and $C$ are three types. Moreover, for each constructor $\kappa$, a $\{0,1\}$-value $\varepsilon(\kappa)$ defines its behaviour with respect to top types (see below).

Types are thus generated through:

$$A, B ::= A \cap B \mid \kappa(\vec{A};\vec{B})$$

Base types are provided by constructors with zero arities.

We introduce a sequent-calculus-style derivation system ISC to define the subtyping relation on these types. We will show that applying proof-theoretical methods, such as cut elimination, allows us to deduce easily some properties of subtyping such as Lemma 10.

Sequents are of the shape $\Gamma \vdash A$ where $\Gamma$ is a (possibly empty) *list* of types. The intended meaning is:

$$A_1, \dots, A_k \vdash B \quad \text{“means”} \quad A_1 \cap \cdots \cap A_k \leq B \qquad\qquad \text{(thus if } k = 0, B \text{ is a top type).}$$

The derivation rules are given in Table 7 and satisfy the subformula property.

**Proposition 5** (Admissible Rules)
*The following rules are admissible in* ISC*:*

$$\dfrac{\Gamma\vdash C}{\Gamma'\vdash C}\,ex \qquad\qquad \dfrac{\Gamma,\Delta\vdash C}{\Gamma,A,\Delta\vdash C}\,wk_{gen} \qquad\qquad \dfrac{}{A\vdash A}\,ax$$

$\Gamma'$ *permutation of* $\Gamma$

$$\dfrac{\Gamma,A\cap B,\Delta\vdash C}{\Gamma,A,B,\Delta\vdash C}\,\cap L_e \qquad\qquad \dfrac{\Gamma,A,A,\Delta\vdash C}{\Gamma,A,\Delta\vdash C}\,co \qquad\qquad \dfrac{\Gamma\vdash A \qquad \Delta,A,\Sigma\vdash C}{\Delta,\Gamma,\Sigma\vdash C}\,cut$$

*Proof.* ($ex$) is obtained by induction on the proof of the premise. ($wk_{gen}$) is obtained by induction on $A$. ($ax$) is obtained by induction on $A$ using ($wk_{gen}$). ($\cap L_e$) is obtained by induction on the premise.

($co$) is obtained by induction on the lexicographically ordered pair (size of $A$, height of the proof of the premise), by looking at each possible last rule of the premise. The key case is ($\cap L$):

$$\dfrac{\Gamma,A,B,A\cap B,\Delta\vdash C}{\Gamma,A\cap B,A\cap B,\Delta\vdash C}\,\cap L$$

we apply ($\cap L_e$) and ($ex$) to the premise to get $\Gamma,A,A,B,B,\Delta\vdash C$ and we use the induction hypothesis twice.

($cut$) is obtained by induction on the lexicographically ordered triple (size of $A$, height of the proof of the left premise, height of the proof of the right premise), by looking at possible last rules of the premises. Let us focus on the main cases:

- ($\cap R$) rule on the right:

$$\dfrac{\begin{array}{c}\pi_1\\ \Gamma\vdash A\end{array} \quad \dfrac{\begin{array}{c}\pi_2\\ \Delta,A,\Sigma\vdash B\end{array} \quad \begin{array}{c}\pi_3\\ \Delta,A,\Sigma\vdash C\end{array}}{\Delta,A,\Sigma\vdash B\cap C}\,\cap R}{\Delta,\Gamma,\Sigma\vdash B\cap C}\,cut \quad\rightsquigarrow\quad \dfrac{\dfrac{\begin{array}{c}\pi_1\\ \Gamma\vdash A\end{array} \quad \begin{array}{c}\pi_2\\ \Delta,A,\Sigma\vdash B\end{array}}{\Delta,\Gamma,\Sigma\vdash B}\,cut \quad \dfrac{\begin{array}{c}\pi_1\\ \Gamma\vdash A\end{array} \quad \begin{array}{c}\pi_3\\ \Delta,A,\Sigma\vdash C\end{array}}{\Delta,\Gamma,\Sigma\vdash C}\,cut}{\Delta,\Gamma,\Sigma\vdash B\cap C}\,\cap R$$

  we use the induction hypothesis twice with a decreasing height on the right.

- ($\cap R$) rule on the left and ($\cap L$) rule on the right:

$$\dfrac{\dfrac{\begin{array}{c}\pi_1\\ \Gamma\vdash A\end{array} \quad \begin{array}{c}\pi_2\\ \Gamma\vdash B\end{array}}{\Gamma\vdash A\cap B}\,\cap R \quad \dfrac{\begin{array}{c}\pi_3\\ \Delta,A,B,\Sigma\vdash C\end{array}}{\Delta,A\cap B,\Sigma\vdash C}\,\cap L}{\Delta,\Gamma,\Sigma\vdash C}\,cut \quad\rightsquigarrow\quad \dfrac{\dfrac{\dfrac{\begin{array}{c}\pi_1\\ \Gamma\vdash A\end{array} \quad \dfrac{\begin{array}{c}\pi_2\\ \Gamma\vdash B\end{array} \quad \begin{array}{c}\pi_3\\ \Delta,A,B,\Sigma\vdash C\end{array}}{\Delta,A,\Gamma,\Sigma\vdash C}\,cut}{\Delta,\Gamma,\Gamma,\Sigma\vdash C}\,cut}{\cdots}\,ex}{\Delta,\Gamma,\Sigma\vdash C}\,co$$

  we use the induction hypothesis twice with smaller cut formulas.

- ($constr$) rules on both sides (in which we only write the key parts):

$$\dfrac{\dfrac{\begin{array}{cccc}\cdots & A_1\vdash A_1^i & \cdots & \ldots,B_1^i,\ldots\vdash B_1\\ & \vdots & & \vdots\\ \cdots & A_\alpha\vdash A_\alpha^i & \cdots & \ldots,B_\beta^i,\ldots\vdash B_\beta\end{array}}{\ldots,\kappa(A_1^i,\ldots,A_\alpha^i;B_1^i,\ldots,B_\beta^i),\ldots\vdash\kappa(A_1,\ldots,A_\alpha;B_1,\ldots,B_\beta)}\,constr \quad \dfrac{\begin{array}{cccc}\cdots & C_1\vdash A_1 & \cdots & \ldots,B_1,\ldots\vdash D_1\\ & \vdots & & \vdots\\ \cdots & C_\alpha\vdash A_\alpha & \cdots & \ldots,B_\beta,\ldots\vdash D_\beta\end{array}}{\ldots,\kappa(A_1,\ldots,A_\alpha;B_1,\ldots,B_\beta),\ldots\vdash\kappa(C_1,\ldots,C_\alpha;D_1,\ldots,D_\beta)}\,constr}{\ldots,\kappa(A_1^i,\ldots,A_\alpha^i;B_1^i,\ldots,B_\beta^i),\ldots\vdash\kappa(C_1,\ldots,C_\alpha;D_1,\ldots,D_\beta)}\,cut$$

$$\rightsquigarrow\quad \dfrac{\cdots \quad \dfrac{C_p\vdash A_p \qquad A_p\vdash A_p^i}{C_p\vdash A_p^i}\,cut \quad \cdots \quad \dfrac{\ldots,B_p^i,\ldots\vdash B_p \qquad \ldots,B_p,\ldots\vdash D_p}{\ldots,B_p^i,\ldots\vdash D_p}\,cut \quad \cdots}{\ldots,\kappa(A_1^i,\ldots,A_\alpha^i;B_1^i,\ldots,B_\beta^i),\ldots\vdash\kappa(C_1,\ldots,C_\alpha;D_1,\ldots,D_\beta)}\,constr$$

  we use the induction hypothesis many times (always with smaller cut formulas). □

Note a 0-ary constructor $\kappa$ behaves like an atomic type (*i.e.* either a type constant or a type variable) if $\varepsilon(\kappa) = 1$, and defines a top type if $\varepsilon(\kappa) = 0$:

$$\dfrac{\dfrac{\varepsilon(\kappa) \leq 0}{\vdash \kappa}\,constr}{A \vdash \kappa}\,wk_{gen}$$

In particular the types obtained with such 0-ary constructors $\kappa$ such that $\varepsilon(\kappa) = 0$ are all equivalent and we denote them $\Omega$. More generally, $\varepsilon(\kappa)$ controls whether $\kappa$ distributes over $\Omega$ or not. In the case of a constructor with unary covariant arity, $\varepsilon(\kappa)$ determines whether $\kappa(\vec{A};\Omega) = \Omega$ or not.

**Proposition 6** (Kernel Properties)
*If we define $A \leq B$ as $A \vdash B$ in* ISC*, the axioms of Table 2 are satisfied.*

*Proof.* (*refl*) and (*trans*) correspond to (*ax*) and (*cut*) from Proposition 5. ($\cap_r$) is an instance of ($\cap R$) and for ($\cap_l^1$) we have:

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\,ax}{A, B \vdash A}\,wk_{gen}}{A \cap B \vdash A}\,\cap L$$

Finally, if we have a 0-ary constructor $\Omega$ with $\varepsilon(\Omega) = 0$, we have just seen it satisfies ($\Omega_r$). □

**Lemma 10** (Inversion)
*If $\kappa(A_1^1,\dots,A_{\alpha_\kappa}^1;B_1^1,\dots,B_{\beta_\kappa}^1),\dots,\kappa(A_1^k,\dots,A_{\alpha_\kappa}^k;B_1^k,\dots,B_{\beta_\kappa}^k) \vdash \kappa(A_1,\dots,A_{\alpha_\kappa};B_1,\dots,B_{\beta_\kappa})$, there exists $\{i_1,\dots,i_p\} \subseteq \{1,\dots,k\}$ such that:*

$$A_1 \vdash A_1^{i_1} \cdots A_1 \vdash A_1^{i_p} \quad \cdots \quad A_{\alpha_\kappa} \vdash A_{\alpha_\kappa}^{i_1} \cdots A_{\alpha_\kappa} \vdash A_{\alpha_\kappa}^{i_p} \quad \mathit{and} \quad B_1^{i_1},\dots,B_1^{i_p} \vdash B_1 \quad \cdots \quad B_{\beta_\kappa}^{i_1},\dots,B_{\beta_\kappa}^{i_p} \vdash B_{\beta_\kappa}$$

*Proof.* By induction on the derivation of $\kappa(A_1^1,\dots,A_{\alpha_\kappa}^1;B_1^1,\dots,B_{\beta_\kappa}^1),\dots,\kappa(A_1^k,\dots,A_{\alpha_\kappa}^k;B_1^k,\dots,B_{\beta_\kappa}^k) \vdash \kappa(A_1,\dots,A_{\alpha_\kappa};B_1,\dots,B_{\beta_\kappa})$, with only (*wk*) and (*constr*) as possible last rules. □

### 3.2 The Arrow-Product Instance

We consider the following set of constructors:

- an at most countable set of 0-ary constructors denoted $X$, $Y$, etc, such that $\varepsilon(X) = \varepsilon(Y) = \cdots = 1$;
- a 0-ary constructor $\Omega$ with $\varepsilon(\Omega) = 0$;
- a constructor $\to$ with contravariant arity 1 and covariant arity 1 such that $\varepsilon(\to) = 0$;
- a constructor $\times$ with contravariant arity 0 and covariant arity 2 such that $\varepsilon(\times) = 1$.

By instantiating the ($constr$) rule of Table 7 to this set of constructors, and using the ($wk$) rule to simplify the $X$ and $\Omega$ cases, we obtain the rules of Table 8 where $k \geq 1$.

**Theorem 1** (Equivalence with BCD)
*$A \vdash B$ in* ISC *with the (constr) rule instantiated as given in Table 8 if and only if $A \leq B$ using the rules of Table 4 extended with the rules of Table 6.*

$$
\frac{}{X \vdash X}\,ax_{at} \qquad \frac{}{\vdash \Omega}\,\Omega \qquad \frac{A \vdash A_1 \quad \cdots \quad A \vdash A_k \quad B_1,\ldots,B_k \vdash B}{A_1 \to B_1,\ldots,A_k \to B_k \vdash A \to B}\,\to \qquad \frac{\vdash B}{\vdash A \to B}\,\to_0
$$

$$
\frac{A_1,\ldots,A_k \vdash A \qquad B_1,\ldots,B_k \vdash B}{A_1 \times B_1,\ldots,A_k \times B_k \vdash A \times B}\,\times
$$

**Table 8:** ISC Deduction System with $\to$ and $\times$.

*Proof.* From left to right, we prove a slightly more general statement: $A_1,\ldots,A_k \vdash B$ implies $\bigcap_{1\leq i\leq k} A_i \leq B$. From right to left, the key results are in Propositions 5 and 6. Main cases are:

$$
\dfrac{\dfrac{\dfrac{}{C \vdash C}\,ax \qquad \dfrac{}{C \vdash C}\,ax \qquad \dfrac{\dfrac{\dfrac{}{A \vdash A}\,ax}{A,B \vdash A}\,wk_{gen} \qquad \dfrac{\dfrac{}{B \vdash B}\,ax}{A,B \vdash B}\,wk_{gen}}{A,B \vdash A \cap B}\,\cap R}{C \to A, C \to B \vdash C \to (A \cap B)}\,\to}{(C \to A) \cap (C \to B) \vdash C \to (A \cap B)}\,\cap L
\qquad
\dfrac{\dfrac{\dfrac{}{\vdash \Omega}\,\Omega}{\vdash \Omega \to \Omega}\,\to_0}{\Omega \vdash \Omega \to \Omega}\,wk
$$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,ax}{A,C \vdash A}\,wk_{gen} \qquad \dfrac{\dfrac{}{C \vdash C}\,ax}{A,C \vdash C}\,wk_{gen}}{A,C \vdash A \cap C}\,\cap R \qquad \dfrac{\dfrac{\dfrac{}{B \vdash B}\,ax}{B,D \vdash B}\,wk_{gen} \qquad \dfrac{\dfrac{}{D \vdash D}\,ax}{B,D \vdash D}\,wk_{gen}}{B,D \vdash B \cap D}\,\cap R}{A \times B, C \times D \vdash (A \cap C) \times (B \cap D)}\,\times}{(A \times B) \cap (C \times D) \vdash (A \cap C) \times (B \cap D)}\,\cap L
$$

□

**Lemma 11** (Inversion for Arrow)
*If $A \leq B$ is obtained from Tables 4 and 6, we have:*

$$
\bigcap_{i\in I}(A_i \to B_i) \leq A \to B \;\Rightarrow\; \exists J \subseteq I,\ \left(\bigcap_{i\in J} B_i \leq B \;\wedge\; \forall i \in J,\ A \leq A_i\right)
$$

This is the key property of subtyping allowing for subject $\beta$-reduction to hold in the BCD typing system. While the traditional proof goes by induction on the derivation which requires a more general statement to deal with the transitivity rule, we rely here on the subformula property. The traditional approach seems more difficult to use in a context where we may have many type constructors.

*Proof.* By Theorem 1, we have $\bigcap_{i\in I}(A_i \to B_i) \vdash A \to B$, thus if $I = \{1,\cdots,k\}$, we get $A_1 \to B_1,\ldots,A_k \to B_k \vdash A \to B$ by Proposition 5. By applying Lemma 10, we obtain $A \vdash A_{i_1},\ldots, A \vdash A_{i_p}$, $B_{i_1},\ldots,B_{i_p} \vdash B$ with $J = \{i_1,\ldots,i_p\} \subseteq I$, so that $\bigcap_{i\in J} B_i \vdash B$, and we conclude with Theorem 1. □

**Lemma 12** (Inversion for Product)
*If $A \leq B$ is obtained from Tables 4 and 6, we have:*

$$
\bigcap_{i\in I}(A_i \times B_i) \leq A \times B \;\Rightarrow\; \bigcap_{i\in I} A_i \leq A \;\wedge\; \bigcap_{i\in I} B_i \leq B
$$

*Proof.* Similarly by Theorem 1, Proposition 5 and Lemma 10. □

### 3.3 BCD Subtyping with Unary Constructors

Our system ISC also generalises BCD subtyping with unary covariant constructors [BCD+18]. In their setting constructors come as a set of unary covariant operations $\kappa$ on types added to the usual $\to$ and $\Omega$ constructors:

$$A, B ::= X \mid A \to B \mid A \cap B \mid \Omega \mid \kappa(A)$$

where each constructor $\kappa$ satisfies the following subtyping properties:

$$\frac{A \leq B}{\kappa(A) \leq \kappa(B)} \qquad\qquad \frac{}{\kappa(A) \cap \kappa(B) \leq \kappa(A \cap B)}$$

This exactly corresponds, in the ISC setting, to a set of constructors $\kappa$ all satisfying $\alpha_\kappa = 0$, $\beta_\kappa = 1$ and $\varepsilon(\kappa) = 1$ (the constructors $\to$ and $\Omega$ are obtained as before). For example the associated (*constr*) rule can be derived in the [BCD+18] setting:

$$\dfrac{\dfrac{}{\bigcap_{1 \leq i \leq k} \kappa(A_i) \leq \kappa(\bigcap_{1 \leq i \leq k} A_i)} \qquad \dfrac{\bigcap_{1 \leq i \leq k} A_i \leq A}{\kappa(\bigcap_{1 \leq i \leq k} A_i) \leq \kappa(A)}}{\bigcap_{1 \leq i \leq k} \kappa(A_i) \leq \kappa(A)}$$

# 4 Conclusion

We have presented a general way of defining a subtyping relation on intersection types which allows us to extend the BCD subtyping to generic contravariant/covariant type constructors. It makes easy to derive key properties used to get subject reduction and subject expansion of the induced type systems. As a concrete example we have fully developed the extension of BCD with product types.

Our approach can be extended to the case where a preorder relation $\preccurlyeq$ between constructors (with the same arities) leads to $\kappa_1 \preccurlyeq \kappa_2 \;\Rightarrow\; \kappa_1(\vec{A};\vec{B}) \leq \kappa_2(\vec{A};\vec{B})$, by a natural generalisation of the (*constr*) rule:

$$\frac{\begin{array}{ccccc} \kappa_1 \preccurlyeq \kappa & A_1 \vdash A_1^1 \;\cdots\; A_1 \vdash A_1^k & B_1^1, \ldots, B_1^k \vdash B_1 & \\ \vdots & \vdots & \vdots & \\ \kappa_k \preccurlyeq \kappa & A_{\alpha_\kappa} \vdash A_{\alpha_\kappa}^1 \;\cdots\; A_{\alpha_\kappa} \vdash A_{\alpha_\kappa}^k & B_{\beta_\kappa}^1, \ldots, B_{\beta_\kappa}^k \vdash B_{\beta_\kappa} & \varepsilon(\kappa) \leq k \end{array}}{\kappa_1(A_1^1, \ldots, A_{\alpha_\kappa}^1; B_1^1, \ldots, B_{\beta_\kappa}^1), \ldots, \kappa_k(A_1^k, \ldots, A_{\alpha_\kappa}^k; B_1^k, \ldots, B_{\beta_\kappa}^k) \vdash \kappa(A_1, \ldots, A_{\alpha_\kappa}; B_1, \ldots, B_{\beta_\kappa})}$$

Ordering constructors is natural in the context of object-oriented languages [KP07, BCD+18].

Another interesting instance would be the study of sum types, but subject expansion looks more complicated. From $\Gamma \vdash u[{}^t/_x] : C$, we can find some type $A$ such that $\Gamma \vdash t : A$ and $\Gamma, x : A \vdash u : C$, but we do not find a way to complete the following derivation:

$$\dfrac{\dfrac{\Gamma \vdash t : A}{\Gamma \vdash \texttt{inl}\ t : A + \_} \qquad \Gamma, x : A \vdash u : C \qquad \dfrac{?}{\Gamma, y : \_ \vdash v : C}}{\Gamma \vdash \begin{array}{l} \texttt{match inl}\ t\ \texttt{with} \\ \mid \texttt{inl}\ x \mapsto u \\ \mid \texttt{inr}\ y \mapsto v \end{array} : C}$$

while $\texttt{match inl}\ t\ \texttt{with} \mid \texttt{inl}\ x \mapsto u \mid \texttt{inr}\ y \mapsto v$ reduces to $u[{}^t/_x]$. The idea of introducing some bottom type $\alpha$ with a rule $\dfrac{}{\Gamma, x : \alpha \vdash t : C}$ seems too naive and breaks the system.

We also plan to work on the characterisation of normalizability properties of terms through typing properties in intersection type systems: solvability, normalization, strong normalization, etc. We would like to extend the known results [BCDC83] to the case with more type constructors.

**Acknowledgements.** We would like to thank to Jan Bessai and Andrej Dudenhefner who suggested investigating BCD with constructors. Thanks also to the anonymous referees for their comments.

## References

[ABDC06] Fabio Alessi, Franco Barbanera & Mariangiola Dezani-Ciancaglini (2006): *Intersection types and lambda models*. Theoretical Computer Science 355(2), pp. 108–126, doi:10.1016/j.tcs.2006.01.004.

[Bak95] Steffen van Bakel (1995): *Intersection Type Assignment Systems*. Theoretical Computer Science 151(2), pp. 385–435, doi:10.1016/0304-3975(95)00073-6.

[BCD+18] Jan Bessai, Tzu-Chun Chen, Andrej Dudenhefner, Boris Düdder, Ugo de Liguoro & Jakob Rehof (2018): *Mixin Composition Synthesis Based on Intersection Types*. Logical Methods in Computer Science 14, p. 37, doi:10.23638/LMCS-14(1:18)2018.

[BCDC83] Henk Barendregt, Mario Coppo & Mariangiola Dezani-Ciancaglini (1983): *A Filter Lambda Model and the Completeness of Type Assignment*. Journal of Symbolic Logic 48, pp. 931–940, doi:10.2307/2273659.

[KP07] Andrew Kennedy & Benjamin Pierce (2007): *On Decidability of Nominal Subtyping with Variance*. In: International Workshop on Foundations and Developments of Object-Oriented Languages (FOOL/WOOD '07). url: `http://foolwood07.cs.uchicago.edu/program/kennedy.pdf` .

[Lau12] Olivier Laurent (2012): *A syntactic introduction to intersection types*. Unpublished note. url: `http://perso.ens-lyon.fr/olivier.laurent/tutinter.pdf` .

[RDRP04] Simona Ronchi Della Rocca & Luca Paolini (2004): *The Parametric Lambda Calculus*. Texts in Theoretical Computer Science, Springer, doi:10.1007/978-3-662-10394-4.

[Ven94] Betti Venneri (1994): *Intersection Types as Logical Formulae*. Journal of Logic and Computation 4(2), pp. 109–124, doi:10.1093/logcom/4.2.109.