A Formulae-as-Types Interpretation of Subtractive Logic

TRISTAN CROLARD, *Laboratory of Algorithmics, Complexity and Logic, Université Paris XII, 61, avenue du Général de Gaulle, 94010 Créteil Cedex, France.*
E-mail: crolard@univ-paris12.fr

Abstract

We present a formulae-as-types interpretation of Subtractive Logic (i.e. bi-intuitionistic logic). This presentation is two-fold: we first define a very natural restriction of the $\lambda\mu$-calculus which is closed under reduction and whose type system is a constructive restriction of the Classical Natural Deduction. Then we extend this deduction system conservatively to Subtractive Logic. From a computational standpoint, the resulting calculus provides a type system for first-class coroutines (a restricted form of first-class continuations).

Keywords: Curry–Howard isomorphism, subtractive logic, control operators, coroutines.

1 Introduction

Subtractive logic, also called 'bi-intuitionistic logic', is an extension of intuitionistic logic with a new connector, the *subtraction* (called *pseudo-difference* in Rauszer's original work [43, 44, 45]), which is dual to implication. This duality has already been widely investigated from algebraic, relational, axiomatic and sequent perspectives by various authors [6, 8, 21, 43–45, 49]. In particular, a unified proof-theoretic approach can be found in Goré's recent paper [21], while a connection with category theory is presented in the author's previous work [6, 8].

We present in this paper a Curry–Howard correspondence for this logic. In other words, we define an extension of the λ-calculus together with its type system such that the logical interpretation of this type system (as a natural deduction system) is exactly subtractive logic. This work is mainly motivated by the following two facts:

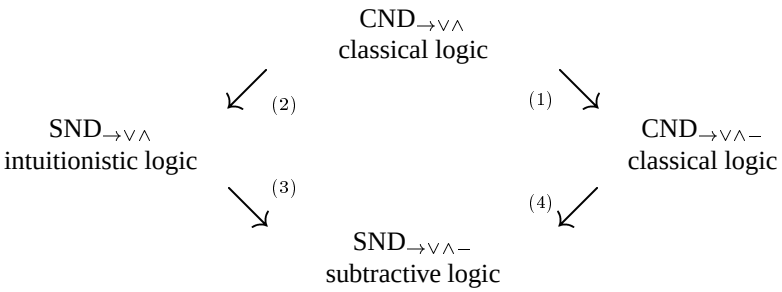
- subtractive logic is not a constructive logic (disjunction and existence properties do not hold in it). In fact, subtractive logic already combines many flavours of classical logic [8]. However, subtractive logic is conservative over some constructive logic (see below). In other words, disjunction and existence properties hold for formulas which contain no subtraction. Therefore, it is likely that a faithful computational interpretation of subtractive logic would shed new light onto the boundary between constructive and classical logics.
- Curien and Herbelin demonstrated in [9] a striking result: duality in classical logic exchanges call-by-value with call-by-name (see also Wadler's recent paper [52]). Actually, a first attempt at a categorical continuation semantics was Filinski's pioneering work [17]. Then Selinger [50] has investigated thoroughly this duality (but still in a categorical setting). On the other hand, Curien and Herbelin's work is based on Parigot's $\lambda\mu$-calculus and its type system, the Classical Natural Deduction (CND) [36, 37]. In order to complete the duality, they are led to extend CND with the subtraction (Walder does not consider the subtraction in [52]). However, this new connective is presented in a purely formal way (without any computational meaning). We believe that

an accurate computational interpretation of subtraction should arise from a formulas-as-types investigation of subtractive logic.

1.1 Subtractive logic

The usual way to define a (sound and faithful) deduction system for subtractive logic is first to add rules for subtraction (which are symmetrical to the rules for implication) and then to restrict sequents to singletons on the left/right sides [21]. Unfortunately, such a calculus is not closed under Parigot's proof normalization process. We shall thus define another restriction which takes into account 'dependencies' between hypotheses and conclusions of a derived sequent. Since these dependencies are defined in a symmetrical manner, our restriction can also be dualized to subtraction. We shall call Subtractive Natural Deduction (SND) the resulting system.

Our method is, on the one hand, to derive introduction and elimination rules for subtraction from its definition in classical logic, and on the other hand, first to restrict the Classical Natural Deduction to intuitionistic logic and then to extend it again to subtractive logic. This process is summarized in the following diagram:



1. Definition of subtraction in classical logic ($A - B \equiv A \wedge \neg B$).

2. Restriction of CND$_{\to\vee\wedge}$ to intuitionistic logic (resp. CDL).

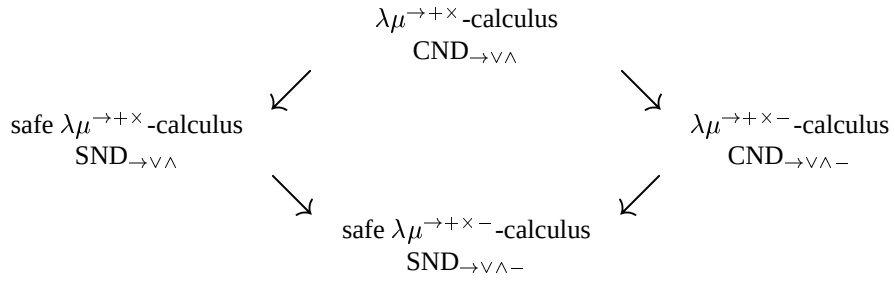
3. Extension of SND$_{\to\vee\wedge}$ with restricted rules for subtraction.

4. Dualized restriction of CND$_{\to\vee\wedge-}$ to subtractive logic.

1.2 Formulae-as-types

The restriction above can be rephrased for the pure (i.e. untyped) $\lambda\mu$-calculus, in such a way that the former restriction is exactly the latter when we consider typed terms. We shall call 'safe $\lambda\mu^{\to+\times-}$-calculus' (resp. 'safe $\lambda\mu^{\to+\times}$-calculus') the subset of restricted $\lambda\mu^{\to+\times-}$-terms (resp. $\lambda\mu^{\to+\times}$-terms). This terminology comes from the computational interpretation of this restriction (see below). The main result of this paper is the proof that these subsets are closed under reduction, which means

that they actually define a calculus. The diagram above is thus completed accordingly:



Let us now rephrase precisely the relation between safe $\lambda\mu^{\rightarrow+\times-}$-terms (resp. safe $\lambda\mu^{\rightarrow+\times}$-terms) and proofs of $\mathrm{SND}_{\rightarrow\vee\wedge-}$ (resp. $\mathrm{SND}_{\rightarrow\vee\wedge}$). This relation can be stated as follows: given the derivation of a typing judgement of a $\lambda\mu^{\rightarrow+\times-}$-term (resp. $\lambda\mu^{\rightarrow+\times}$-term) t by some sequent in $\mathrm{CND}_{\rightarrow\vee\wedge-}$ (resp. $\mathrm{CND}_{\rightarrow\vee\wedge}$), if t is safe then this derivation belongs to $\mathrm{SND}_{\rightarrow\vee\wedge-}$ (resp. $\mathrm{SND}_{\rightarrow\vee\wedge}$). As a consequence, the subject reduction for $\mathrm{CND}_{\rightarrow\vee\wedge-}$ (resp. $\mathrm{CND}_{\rightarrow\vee\wedge}$) together with the closure under reduction of the safe $\lambda\mu^{\rightarrow+\times-}$-calculus (resp. $\lambda\mu^{\rightarrow+\times}$-calculus) provides the subject reduction for $\mathrm{SND}_{\rightarrow\vee\wedge-}$ (resp. $\mathrm{SND}_{\rightarrow\vee\wedge}$).

1.3 Strong normalization and the subformula property

Since our calculi are derived from de Groote's $\lambda\mu^{\rightarrow\wedge\vee\perp}$, they enjoy the properties listed in [12], namely:

(a) all connectives are taken as primitive;

(b) normal deductions satisfy the subformula property;

(c) the reduction relation is defined by means of local reduction steps;

(d) the reduction relation is strongly normalizing ;

(e) the reduction relation is Church–Rosser;

(f) the reduction relation is defined at the untyped level;

(g) the reduction relation satisfies the subject reduction property.

In [43], C. Rauszer defined a sequent calculus for subtractive logic which enjoys cut-elimination (and the subformula property which follows). As a by-product of the previous properties, we obtain a new proof of this result (actually, we prove a stronger result, namely the *strong* normalization). As a corollary of the subformula property, we obtain that the normal form of the deduction of a sequent in $\mathrm{CND}_{\rightarrow\vee\wedge-}$ (resp. $\mathrm{SND}_{\rightarrow\vee\wedge-}$) which does not contain any occurrence of subtraction belongs to $\mathrm{CND}_{\rightarrow\vee\wedge}$ (resp. $\mathrm{SND}_{\rightarrow\vee\wedge}$).

1.4 Computational interpretation

Since Griffin's pioneering work [23], the extension of the well-known formulas-as-types paradigm to classical logic has been widely investigated for instance by Murthy [31], Barbanera and Berardi [2], Rehof and Sørensen [46], de Groote [11, 12], Krivine [30]. We shall consider here Parigot's $\lambda\mu$-calculus mainly because it is confluent and strongly normalizing in the second-order framework [38]. However, Parigot's original CND is a second-order logic, in which $\vee, \wedge, \exists, \exists^2$ are definable from $\rightarrow, \forall, \forall^2$. Since we are also interested in the subformula property, we shall restrict ourselves

to the propositional framework where any connective is taken as primitive and where the proof normalization process includes permutative conversions [12]. Such an extension of CND with primitive conjunction and disjunction has already been investigated by Pym, Ritter and Wallen [40, 41, 39] and de Groote [12].

The computational interpretation of classical logic is usually given by a λ-calculus extended with some form of control (such as the famous **call/cc** of Scheme or the catch/throw mechanism of Lisp) or similar formulations of first-class continuation constructs. Continuations are used in denotational semantics to describe control commands such as jumps. They can also be used as a programming technique to simulate backtracking and coroutines. For instance, first-class continuations have been successfully used to implement Simula-like cooperative coroutines in Scheme [53, 18, 19]. This approach has been extended in the Standard ML of New Jersey (with the typed counterpart of Scheme's **call/cc** [15]) to provide simple and elegant implementations of light-weight processes (or threads), where concurrency is obtained by having individual threads voluntarily suspend themselves [48, 42] (providing time-sliced processes using pre-emptive scheduling requires additional run-time system support [5, 47]). The key point in these implementations is that control operators make it possible to switch between coroutine contexts, where the context of a coroutine is encoded as its continuation.

The definition of a catch/throw mechanism is straightforward in the $\lambda\mu$-calculus: just set **catch** $\alpha\ t \equiv \mu\alpha[\alpha]t$ and **throw** $\alpha\ t \equiv \mu\delta[\alpha]t$ where δ is a name which does not occur in t (see [7] for a study of the sublanguage obtained when we restrict the $\lambda\mu$-terms to these operators). Then a name α may be reified as the first-class continuation $\lambda x.$**throw** $\alpha\ x$. However, the type of such a $\lambda\mu$-term is the excluded-middle $\vdash A^{\alpha}; \neg A$. Thus, a continuation cannot be a first-class citizen in a constructive logic. To figure out what kind of restricted use of continuations is allowed in SND $_{\rightarrow\vee\wedge}$, we observe that in the restricted $\lambda\mu^{\rightarrow+\times}$-calculus, even if continuations are no longer first-class objects, the ability of context-switching remains (in fact, this observation is easier to make in the framework of abstract state machines). However, a context is now a pair $\langle$*environment*, *continuation*$\rangle$. Note that such a pair is exactly what we expect as the context of a coroutine, since a coroutine should not access the local environment (the part of the environment which is not shared) of another coroutine. Consequently, we say that a $\lambda\mu^{\rightarrow+\times}$-term t is 'safe with respect to coroutine contexts' (or just 'safe' for short) if no coroutines of t access the local environment of another coroutine.

The extension of this interpretation to SND$_{\rightarrow\vee\wedge-}$ is now straightforward. We already know (from its definition in classical logic) that subtraction is a good candidate to type a pair $\langle$*environment*, *continuation*$\rangle$. SND$_{\rightarrow\vee\wedge-}$ is thus interpreted as a type system for first-class coroutines. The dual restriction from the safe $\lambda\mu^{\rightarrow+\times-}$-calculus just ensures that one cannot obtain full-fledged first-class continuations back from first-class coroutines.

1.5 Related work

Nakano, Kameyama and Sato [26–28, 32–34] have proposed various logical frameworks that are intended to provide a type system for a lexical variant of the catch/throw mechanism used in functional languages such as Lisp. Moreover, Nakano has shown that it is possible to restrict the catch/throw mechanism in order to stay in an intuitionistic (propositional) framework. However, in their approach, a disjunction is used to type first-class exceptions. In this paper, we generalize these results in several ways:

- We use Parigot's $\lambda\mu$-calculus and its type system, the Classical Natural deduction which is confluent and strongly normalizing in the second-order framework [38].
- We consider a type system *à la* Curry, which allows us to rephrase the above restriction on pure

(*i.e.* untyped) $\lambda\mu$-terms, and not only on typed terms as in [34].

Brauner and de Paiva [4] have proposed a restriction of Classical Linear Logic (in order to obtain a sequent-style formulation of Full Intuitionistic Linear Logic defined by Hyland and de Paiva [25]) very akin to the one presented in this paper. In a recent work [13], de Paiva and Ritter also consider a Parigot-style linear λ-calculus for this logic which is based on Pym and Ritter's $\lambda\mu\nu$-calculus [41]. If we forget about linearity (which has of course many consequences on the resulting system), the main advantages of our approach are the following ones:

- We propose a computational interpretation of our restriction (as 'coroutines which do not access the local environment on another coroutine').
- Our restriction is symmetrical and thus easily extends to duality. This allows us to propose a computational interpretation of subtraction (as a type constructor for 'first-class coroutines').

1.6 About first-order subtractive logic

Propositional subtractive logic is conservative over intuitionistic logic. However, in the first-order framework, subtractive logic is no longer conservative over intuitionistic logic but over Constant Domain Logic (CDL). This logic has been studied for instance in [20, 22, 35] and [45]. Although CDL is not conservative over intuitionistic logic (in the first-order framework), it is important to note that CDL is still a constructive logic (i.e. both disjunction and existence properties hold in it [22]). Moreover, CDL if fully axiomatized as the first-order intuitionistic logic extended with the following axiom schema (called DIS in [45]) where x does not occur in B (see [20, 35] for instance):

$$\forall x(A \vee B) \vdash \forall x A \vee B.$$

Recall however that this paper is devoted to the computational interpretation of *propositional* subtractive logic.

1.7 Plan of the paper

The remainder of the paper is organized as follows. In Section 2, we present $\mathrm{CND}_{\to\vee\wedge-}$ which is our extension of $\mathrm{CND}_{\to\vee\wedge}$ with the subtraction. In Section 3, we present the $\mathrm{SND}_{\to\vee\wedge-}$ which is our restriction of $\mathrm{CND}_{\to\vee\wedge-}$ to subtractive logic. In Section 4, we recall the $\lambda\mu^{\to+\times}$-calculus and we introduce the $\lambda\mu^{\to+\times-}$-calculus. In Section 5, we define the safe $\lambda\mu^{\to+\times-}$-calculus and we show how it is related to $\mathrm{CND}_{\to\vee\wedge-}$. Eventually, in Section 6, we prove that the safe $\lambda\mu^{\to+\times-}$-calculus is closed under reduction.

2 The system $\mathrm{CND}_{\to\vee\wedge-}$

We consider sequents with the form $\Gamma \vdash \Delta; A$ where Γ, Δ are sets of *named* propositional formulas. As usual, the semi-colon serves to single out one distinguished (or 'active') conclusion of a sequent, which clearly amounts to naming this occurrence with a special name (when needed, we shall use the name '$[]$' for this 'unnamed' occurrence).

The deductions rules for $\mathrm{CND}_{\to\vee\wedge-}$ (CND with conjunction, disjunction, subtraction and an explicit cut rule) are given in Figure 1. We write '$\Delta(, A^{\alpha})$' in the premiss of the *activate/passivate* rules to outline the fact that A^{α} is allowed but not required in the succedent of the sequent. Note that if A does not occur in Δ in the premiss of the *activate* rule the we obtain a weakening rule. Similarly,

$$A \vdash A \ (\textit{axiom})$$

$$\frac{\Gamma, A, A \vdash \Delta; B}{\Gamma, A \vdash \Delta; B}(C_L) \qquad \frac{\Gamma \vdash \Delta; B}{\Gamma, A \vdash \Delta; B}(W_L)$$

$$\frac{\Gamma \vdash \Delta(, A^\alpha); }{\Gamma \vdash \Delta; A}(\textit{activate}) \qquad \frac{\Gamma \vdash \Delta(, A^\alpha); A}{\Gamma \vdash \Delta, A^\alpha; }(\textit{passivate})$$

$$\frac{\Gamma \vdash \Delta; A \quad \Gamma, A \vdash \Delta; B}{\Gamma \vdash \Delta; B}(\textit{cut})$$

$$\frac{\Gamma, A \vdash \Delta; B}{\Gamma \vdash \Delta; A \to B}(I_\to) \qquad \frac{\Gamma \vdash \Delta; A \to B \quad \Gamma \vdash \Delta; A}{\Gamma \vdash \Delta; B}(E_\to)$$

$$\frac{\Gamma \vdash \Delta; A \quad \Gamma \vdash \Delta; B}{\Gamma \vdash \Delta; A \wedge B}(I_\wedge)$$

$$\frac{\Gamma \vdash \Delta; A \wedge B}{\Gamma \vdash \Delta; A}(E^1_\wedge) \qquad \frac{\Gamma \vdash \Delta; A \wedge B}{\Gamma \vdash \Delta; B}(E^2_\wedge)$$

$$\frac{\Gamma \vdash \Delta; A}{\Gamma \vdash \Delta; A \vee B}(I^1_\vee) \qquad \frac{\Gamma \vdash \Delta; B}{\Gamma \vdash \Delta; A \vee B}(I^2_\vee)$$

$$\frac{\Gamma \vdash \Delta; A \vee B \quad \Gamma, A \vdash \Delta; C \quad \Gamma, B \vdash \Delta; C}{\Gamma \vdash \Delta; C}(E_\vee)$$

$$\frac{\Gamma \vdash \Delta(, C^\gamma); A \quad \Gamma, B \vdash \Delta(, C^\gamma); C}{\Gamma \vdash \Delta, C^\gamma; A - B}(I_-)$$

$$\frac{\Gamma \vdash \Delta; A - B \quad \Gamma, A \vdash \Delta; B}{\Gamma \vdash \Delta; C}(E_-)$$

FIGURE 1. System $\mathrm{CND}_{\to\vee\wedge-}$

if A^α already occurs in Δ in the conclusion of the *passivate* rule then we obtain a contraction rule. Moreover, the *activate/passivate* rules allow us to derive the following weakening, contraction and exchange rules (on the right-hand side):

$$\frac{\Gamma \vdash \Delta(, A^\alpha); A}{\Gamma \vdash \Delta; A}(C_R) \qquad \frac{\Gamma \vdash \Delta(, A^\alpha); A}{\Gamma \vdash \Delta, A^\alpha; B}(W_R) \qquad \frac{\Gamma \vdash \Delta(, B^\beta)(, A^\alpha); B}{\Gamma \vdash \Delta, B^\beta; A}(Ex_R)$$

Note that we may replace the semi-colon by a comma in the rules whenever we want to allow implicit exchange (and consider named conclusions up to permutation).

Let us now comment on the introduction/elimination rules for subtraction. The introduction rule is dual to the left-hand side introduction rule for implication (in LK), while the elimination rule is reminiscent of the elimination rule for disjunction. A simpler introduction rule for the subtraction would be the following one:

$$\frac{\Gamma \vdash \Delta(, B^\beta); A}{\Gamma \vdash \Delta, B^\beta; A - B}(I'_-)$$

This rule is equivalent to (I_-). Indeed, (I'_-) is derivable from (I_-) as follows (for the sake of

simplicity, we omit Γ,Δ in axioms):

$$\frac{\Gamma \vdash \Delta(, B^{\beta}); A \quad B \vdash ; B}{\Gamma \vdash \Delta, B^{\beta}; A - B}(I_{-})$$

Conversely, (I_{-}) is derivable from (I'_{-}) using the *cut* rule (we omit the structural rules):

$$\frac{\dfrac{\Gamma \vdash \Delta(, C^{\gamma}), A}{\Gamma \vdash \Delta(, C^{\gamma}), B^{\beta}, A - B}(I'_{-}) \qquad \Gamma, B \vdash \Delta(, C^{\gamma}), C}{\Gamma \vdash \Delta, C^{\gamma}, A - B}(\mathit{cut})$$

We shall give more details about the differences between (I_{-}) and (I'_{-}) in Section 4.

2.1 *Defining $A - B$ as $A \wedge \neg B$*

Since $A - B$ is definable in classical logic as $A \wedge \neg B$, let us prove that the introduction/elimination rules for subtraction are derivable. In order to deal with the negation, we add the propositional constant for *falsum* $\bot$. We use the same name ε for occurrences of $\bot$ in all sequents. Moreover, we do not represent $\bot^{\varepsilon}$ in the conclusions of sequents (see [1] for more details about the various treatments of $\bot$ in CND). Now the elimination rule for $\bot$ is just an instance of (W_R) (take $\bot^{\varepsilon}$ as A^{α}):

$$\frac{\Gamma \vdash \Delta; \bot}{\Gamma \vdash \Delta; B}(E_{\bot})$$

As usual, we also define $\neg A$ as $(A \to \bot)$.

Derivation of the introduction rule (I'_{-})

$$\frac{\Gamma \vdash \Delta; A \qquad \dfrac{\dfrac{\Gamma, B \vdash \Delta; B}{\Gamma \vdash \Delta, B; \bot}(W_R)}{\Gamma \vdash \Delta, B; \neg B}(I_{\to})}{\Gamma \vdash \Delta, B; A \wedge \neg B}(I_{\wedge})$$

Derivation of the elimination rule (E_{-})

$$\frac{\Gamma \vdash \Delta; A \wedge \neg B \qquad \dfrac{\dfrac{\neg B \vdash \neg B \quad \Gamma, A \vdash \Delta; B}{\Gamma, A, \neg B \vdash \Delta; \bot}(E_{\to})}{\Gamma, A, \neg B \vdash \Delta; C}(E_{\bot})}{\Gamma \vdash \Delta; C}(E_{\wedge})$$

where the last rule $(E_{\wedge})$ is easily derivable from $(E^1_{\wedge})$, $(E^2_{\wedge})$ and (*cut*).

2.2 *About the duality*

The rules chosen for disjunction (resp. subtraction) in CND $_{\to \vee \wedge -}$ are clearly not dual to the rules for conjunction (resp. implication). As expected, such rules are left-hand side introduction/elimination rules. For instance, the rules for disjunction which are dual to the rules for conjunction are the following ones:

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}(\mathit{LI}_{\vee}) \qquad \frac{\Gamma, A \vee B \vdash \Delta}{\Gamma, A \vdash \Delta}(\mathit{LE}^{1}_{\vee}) \qquad \frac{\Gamma, A \vee B \vdash \Delta}{\Gamma, B \vdash \Delta}(\mathit{LE}^{2}_{\vee})$$

It is easy to show that these rules are equivalent to the (right-hand side) introduction/elimination for disjunction. Similarly, we could define by duality left-hand side introduction/elimination rules for subtraction:

$$\frac{\Gamma, A \vdash \Delta, B}{\Gamma, A - B \vdash \Delta}(LI_{-}) \qquad \frac{\Gamma, A - B \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vdash \Delta}(LE_{-})$$

Let us prove that these rules are indeed equivalent to the (right-hand side) introduction/elimination rules:

Derivation of the left-hand side introduction rule (LI_{-})

$$\frac{\dfrac{A - B \vdash A - B \qquad \Gamma, A \vdash \Delta, B}{\Gamma, A - B \vdash \Delta, \bot}(E_{-})}{\Gamma, A - B \vdash \Delta}(C_R)$$

Derivation of the left-hand side elimination rule (LE_{-})

$$\frac{\dfrac{A \vdash A \quad \Gamma, B \vdash \Delta}{\Gamma, A \vdash \Delta, A - B}(I_{-}) \qquad \Gamma, A - B \vdash \Delta}{\Gamma, A \vdash \Delta}(cut)$$

Conversely, the (right-hand side) introduction/elimination rules are derivable from the left-hand side rules. Indeed:

Derivation of the introduction rule (I_{-})

$$\frac{\Gamma \vdash \Delta, A \qquad \dfrac{A - B \vdash A - B \quad \Gamma, B \vdash \Delta, C}{\Gamma, A \vdash \Delta, C, A - B}(LE_{-})}{\Gamma \vdash \Delta, C, A - B}(cut)$$

Derivation of the elimination rule (E_{-})

$$\frac{\dfrac{\Gamma \vdash \Delta, A - B \qquad \dfrac{\Gamma, A \vdash \Delta, B}{\Gamma, A - B \vdash \Delta}(LE_{-})}{\Gamma \vdash \Delta}(cut)}{\Gamma \vdash \Delta, C}(W_R)$$

DEFINITION 2.1
We call Symmetric CND$_{\rightarrow\vee\wedge-}$ the system CND$_{\rightarrow\vee\wedge-}$ where the rules for disjunction and subtraction are replaced by $(LI_{\vee})$, $(LE^{1}_{\vee})$, $(LE^{2}_{\vee})$, and (LI_{-}), (LE_{-}).

3 Constructive restrictions of CND

It is well known that if we restrict the classical sequent calculus LK [51] to sequents with at most one conclusion we obtain the intuitionistic sequent calculus LJ [51]. As for natural deduction, it was originally presented for sequents having one conclusion and formalized intuitionistic logic. Parigot's CND may be seen as an extension of natural deduction to sequents with several conclusions. As expected, this extension leads to classical logic. In order to stay in a constructive framework, several authors (for instance [29] p. 481 and also [3, 14, 16]) have suggested restricting only the introduction rule of implication of LK to sequents with at most one conclusion. The same restriction can be applied to CND and by duality it can be generalized to Symmetric CND$_{\rightarrow\vee\wedge-}$ (Definition 2.1) as follows:

DEFINITION 3.1
We call Symmetric $\mathrm{SND}^1_{\to\vee\wedge-}$ the system Symmetric $\mathrm{CND}_{\to\vee\wedge-}$ where:

1. the rule $(I_\to)$ is restricted to sequents with at most one conclusion,
2. the rule (LI_-) is restricted to sequents with at most one hypothesis.

PROPOSITION 3.2
Symmetric $\mathrm{SND}^1_{\to\vee\wedge-}$ is sound and complete for Subtractive Logic.

PROOF. By induction on the derivation, we prove that a sequent $\Gamma_1, \ldots, \Gamma_m \vdash \Delta_1, \ldots, \Delta_n$ is derivable in Symmetric $\mathrm{SND}^1_{\to\vee\wedge-}$ iff $\Gamma_1 \wedge \ldots \wedge \Gamma_m \vdash \Delta_1 \vee \ldots \vee \Delta_n$ is derivable in the symmetrical propositional categorical calculus defined in [8]. ∎

REMARK 3.3
The restriction on the rule (LI_-) is required, otherwise we are still in classical logic (even if the introduction rule for implication is restricted). See [8] for more details.

However, this restriction of the introduction rule for implication is not closed under Parigot's proof normalization process. Let us consider for instance the following proof in CND where $(I_\to)$ is applied only to sequents with at most one conclusion:

$$\dfrac{\dfrac{\dfrac{\dfrac{B \vdash ; B}{A, B \vdash ; B}(W_L)}{B \vdash ; A \to B}(I_\to)}{\vdash ; B \to (A \to B)}(I_\to) \quad \dfrac{\vdots}{\Gamma \vdash \Delta ; B}}{\Gamma \vdash \Delta ; A \to B}(E_\to)$$

Since $(I_\to)$ is immediately followed by the last $(E_\to)$, this proof may be replaced by:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash \Delta ; B}}{\Gamma, A \vdash \Delta ; B}(W_L)}{\Gamma \vdash \Delta ; A \to B}(I_\to)$$

And in this proof, the rule $(I_\to)$ is applied to a sequent with multiple conclusions. Note that we could try to detect when multiple conclusions are harmless (from the constructive standpoint) in an occurrence of $(I_\to)$. This was also observed by Ritter, Pym and Wallen in [41], where a notion of 'superfluous subderivations' is defined. Since their definition relies on weakening occurrences in proof terms, we shall come back to this question in Section 5 (Remark 5.13). We shall here consider another restriction of CND which enjoys the following properties: it is closed under proof normalization, it is expressed independently of proof terms and it is designed to easily extend by symmetry to SND.

Our restriction is based on dependencies between occurrences of hypotheses and occurrences of conclusions in a derived sequent, where dependencies come from axioms and are propagated by inference rules. Then some hypothesis may be discharged onto some conclusion if and only if no other conclusion depends on it.

A very similar notion of dependency, and the same restriction was discovered independently by Brauner and de Paiva [4, 25], and applied to the definition of Full Intuitionistic Linear Logic. In a recent work [13], de Paiva and Ritter also present a Parigot-style linear λ-calculus for this logic which is based on Pym and Ritter's $\lambda\mu\nu$-calculus.

Note however that linearity has many consequences on the resulting system. For instance, the properties of their multiplicative 'disjunction' (the connective $\Box$) are very different from the properties of our (usual) intuitionistic disjunction. Moreover we do not need any specific typed λ-calculus in order to track how dependencies are propagated (as in [13]). Our annotations are just sets of names.

Eventually, a major difference (if we forget about linearity) with their work in [4, 25] is that we emphasize here the symmetry of our definition of dependency relations in order to take the subtraction into account.

3.1 A restriction of CND based on dependency relations

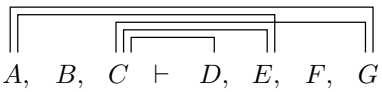
We use undirected links to make explicit all *dependencies* between occurrences of hypotheses and occurrences of conclusions in any derived sequent. A link between some occurrence of a hypothesis Γ_i and some occurrence of a conclusion Δ_j may be represented as follows:

$$\Gamma_1 \,, \ldots\ \Gamma_i\ , \ldots\ \Gamma_n\ \vdash\ \Delta_1 \,, \ldots\ \Delta_j\ , \ldots \Delta_m$$

In order to annotate sequents with such links, we name any *occurrence* of hypotheses ($x, y, z, \ldots$) and any *occurrence* of conclusions ($\alpha, \beta, \gamma, \ldots$) in a sequent. We assume that the name of a hypothesis (resp. a conclusion) never occurs twice in a sequent. The links annotating some sequent $\Gamma_1^{x_1}, \ldots, \Gamma_n^{x_n} \vdash \Delta_1^{\alpha_1}, \ldots, \Delta_m^{\alpha_m}$ provide a subset of $\{x_1, \ldots, x_n\} \times \{\alpha_1, \ldots, \alpha_m\}$ which can be represented by annotating either each conclusion by a set of hypotheses or each hypothesis by a set of conclusions.

EXAMPLE 3.4

Consider the sequent $A^x, B^y, C^z \vdash D^\alpha, E^\beta, F^\gamma, G^\delta$ together with the dependencies $\{(x, \beta), (x, \delta), (z, \alpha), (z, \beta), (z, \delta)\}$:



This annotated sequent can be represented in both forms:

- each conclusion is annotated by a set of hypotheses

$$A^x, B^y, C^z \vdash \{z\} : D, \{x, z\} : E, \{\} : F, \{x, z\} : G.$$

- each hypothesis is annotated by a set of conclusions

$$\{\beta, \delta\}{:}A, \{\} : B, \{\alpha, \beta, \delta\} : C \vdash D^\alpha, E^\beta, F^\gamma, G^\delta.$$

3.1.1 Annotations for Symmetric CND$_{\rightarrow\vee\wedge-}$

We first present the annotations for the multiplicative context variant of Symmetric CND$_{\rightarrow\vee\wedge-}$. Multiplicative rules are easier to annotate since the dependencies in the context are not modified by the rule. Thus, in these multiplicative rules, we assume that a formula does not occur with the same name in both Γ and Γ' (resp. Δ and Δ'). Note that we use here both representations of dependencies (i.e. in some rules conclusions are annotated while in dual rules, hypotheses are annotated) in order to emphasize the symmetry of the definition. The annotated rules of Symmetric CND$_{\rightarrow\vee\wedge-}$ are summarized in Figure 2.

$$A^x \vdash \{x\} : A \ (\textit{axiom})$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, A^x \vdash \Delta}(W_L) \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta,\ \{\} : A}(W_R)$$

$$\frac{\Gamma,\ U : A,\ V : A \vdash \Delta}{\Gamma,\ U \cup V : A \vdash \Delta}(C_L) \qquad \frac{\Gamma \vdash \Delta,\ U : A,\ V : A}{\Gamma \vdash \Delta,\ U \cup V : A}(C_R)$$

$$\frac{\Gamma \vdash \Delta,\ S : A \quad \Gamma', A^x \vdash S'_1 : \Delta'_1, \ldots,\ S'_p : \Delta'_p}{\Gamma, \Gamma' \vdash \Delta, S'_1[S/x] : \Delta'_1, \ldots,\ S'_p[S/x] : \Delta'_p}(\textit{cut})$$

$$\frac{\Gamma, A^x \vdash S_1 : \Delta_1, \ldots,\ S_n : \Delta_n,\ S : B}{\Gamma \vdash S_1 \setminus \{x\} : \Delta_1, \ldots,\ S_n \setminus \{x\} : \Delta_n,\ S \setminus \{x\} : (A \to B)}(I_\to)$$

$$\frac{\Gamma \vdash \Delta,\ U : (A \to B) \quad \Gamma' \vdash \Delta',\ V : A}{\Gamma, \Gamma' \vdash \Delta, \Delta',\ U \cup V : B}(E_\to)$$

$$\frac{\Gamma \vdash \Delta, S : A \wedge B}{\Gamma \vdash \Delta, S : A}(E^1_\wedge) \qquad \frac{\Gamma \vdash \Delta, S : A \wedge B}{\Gamma \vdash \Delta, S : B}(E^2_\wedge)$$

$$\frac{\Gamma \vdash \Delta, U : A \quad \Gamma' \vdash \Delta', V : B}{\Gamma, \Gamma' \vdash \Delta, \Delta', U \cup V : A \wedge B}(I_\wedge)$$

$$\frac{\Gamma, S : A \vee B \vdash \Delta}{\Gamma, S : A \vdash \Delta}(LE^{\ 1}_{\ \vee}) \quad \frac{\Gamma, S : A \vee B \vdash \Delta}{\Gamma, S : B \vdash \Delta}(LE^{\ 2}_{\ \vee})$$

$$\frac{\Gamma,\ U : A \vdash \Delta \quad \Gamma',\ V : B \vdash \Delta'}{\Gamma, \Gamma',\ U \cup V : A \vee B \vdash \Delta, \Delta'}(LI_{\ \vee})$$

$$\frac{\Gamma, U : A - B \vdash \Delta \quad \Gamma', V : B \vdash \Delta'}{\Gamma, \Gamma', U \cup V : A \vdash \Delta, \Delta'}(LE_-)$$

$$\frac{S_1 : \Gamma_1, \ldots,\ S_m : \Gamma_m, S : A \vdash \Delta, B^\beta}{S_1 \setminus \{\beta\} : \Gamma_1, \ldots,\ S_m \setminus \{\beta\} : \Gamma_m, S \setminus \{\beta\} : A - B \vdash \Delta}(LI_-)$$

FIGURE 2. Annotations for Symmetric CND$_{\to\vee\wedge-}$

NOTE 3.5
We shall use the following abbreviation:

$$U[V/x] \equiv \begin{cases} U \setminus \{x\} \cup V \text{ if } x \in U \\ U \text{ otherwise.} \end{cases}$$

We also omit annotations which are not modified by a rule.

3.1.2 Restriction of Symmetric $\mathrm{CND}_{\to\vee\wedge-}$ to Subtractive Logic

DEFINITION 3.6
In an annotated proof of Symmetric $\mathrm{CND}_{\to\vee\wedge-}$, we say that an occurrence of the rule $(I_{\to})$ is *constructive* iff x does not occur in any S_i. We obtain then:

$$\frac{\Gamma, A^x \vdash S_1 : \Delta_1, \ldots,\ S_n : \Delta_n,\ V : B}{\Gamma \vdash S_1 : \Delta_1, \ldots,\ S_n : \Delta_n,\ V \setminus \{x\} : (A \to B)} \quad \text{where } x \notin S_1 \cup \ldots \cup S_n.$$

In other words, to stay in a constructive framework, a hypothesis may be discharged onto some conclusion if and only if no other conclusion depends on it. By duality, we obtain the following restriction on (LI_{-}):

DEFINITION 3.7
In an annotated proof of Symmetric $\mathrm{CND}_{\to\vee\wedge-}$, we say that an occurrence of the rule (LI_{-}) is *constructive* iff β does not occur in any S_i. We obtain then:

$$\frac{S_1 : \Gamma_1, \ldots,\ S_m : \Gamma_m, S : A \vdash \Delta, B^{\beta}}{S_1 : \Gamma_1, \ldots,\ S_m : \Gamma_m, S \setminus \{\beta\} : A - B \vdash \Delta} \quad \text{where } \beta \notin S_1 \cup \ldots \cup S_m\ .$$

DEFINITION 3.8
We say that a proof of Symmetric $\mathrm{CND}_{\to\vee\wedge-}$ is *constructive* iff:

1. any occurrence $(I_{\to})$ is *constructive*,
2. any occurrence (LI_{-}) is *constructive*.

DEFINITION 3.9
We call Symmetric $\mathrm{SND}_{\to\vee\wedge-}$ the restriction of Symmetric $\mathrm{CND}_{\to\vee\wedge-}$ to constructive proofs.

THEOREM 3.10
Symmetric $\mathrm{SND}_{\to\vee\wedge-}$ is sound and complete for Subtractive Logic.

PROOF. (sketch) It is clear that Symmetric $\mathrm{SND}_{\to\vee\wedge-}$ is a generalization of Symmetric $\mathrm{SND}^1_{\to\vee\wedge-}$, thus it is sufficient to prove that any sequent derivable in Symmetric $\mathrm{SND}_{\to\vee\wedge-}$ is also derivable in Symmetric $\mathrm{SND}^1_{\to\vee\wedge-}$. The idea of the proof is the following one: any derivation of a sequent in Symmetric $\mathrm{SND}_{\to\vee\wedge-}$ can be translated into a derivation which does not contain any introduction rule of implication nor any left-hand side introduction rule of subtraction, but which depends only on axioms valid in Symmetric $\mathrm{SND}^1_{\to\vee\wedge-}$. For that purpose, we show that (a generalized version of) constructive $(I_{\to})$ 'commutes' with every other rule of Symmetric $\mathrm{SND}_{\to\vee\wedge-}$. By duality, we know that (a generalized version of) constructive (LI_{-}) also 'commutes' with every other rule of Symmetric $\mathrm{SND}_{\to\vee\wedge-}$. There is thus a procedure which removes occurrences of constructive $(I_{\to})$ and constructive (LI_{-}) in the original proof and yields a proof of Symmetric $\mathrm{SND}^1_{\to\vee\wedge-}$. The details are given in Appendix B. ∎

3.2 Annotations for $\mathrm{CND}_{\to\vee\wedge-}$

We derive here the annotations for $\mathrm{CND}_{\to\vee\wedge-}$ from the annotations of Symmetric $\mathrm{CND}_{\to\vee\wedge-}$. There are several steps to perform:

- derive rules with additive contexts (using the weakening and contraction rules),
- move all the annotations on the right-hand side (each conclusion is annotated by a set of hypotheses),

$$A^x \vdash ; \{x\} : A \ (\textit{axiom})$$

$$\frac{\Gamma,\ U : A,\ V : A \vdash \Delta; B}{\Gamma,\ U \cup V : A \vdash \Delta; B}(C_L) \qquad \frac{\Gamma \vdash \Delta; B}{\Gamma, A^x \vdash \Delta; B}(W_L)$$

$$\frac{\Gamma \vdash \Delta(, U : A^\alpha);}{\Gamma \vdash \Delta; U : A}(\textit{activate}) \qquad \frac{\Gamma \vdash \Delta(, U : A^\alpha); V : A}{\Gamma \vdash \Delta, U \cup V : A^\alpha;}(\textit{passivate})$$

$$\frac{\Gamma \vdash S_1 : \Delta_1, \ldots, S_n : \Delta_n;\ S : A \quad \Gamma, A^x \vdash S'_1 : \Delta_1, \ldots,\ S'_n : \Delta_n; S' : B}{\Gamma \vdash S_1 \cup S'_1[S/x] : \Delta_1, \ldots,\ S_n \cup S'_n[S/x] : \Delta_n; S'[S/x] : B}(\textit{cut})$$

$$\frac{\Gamma, A^x \vdash S_1 : \Delta_1, \ldots,\ S_n : \Delta_n; S : B}{\Gamma \vdash S_1 \setminus \{x\} : \Delta_1, \ldots,\ S_n \setminus \{x\} : \Delta_n; S \setminus \{x\} : (A \to B)}(I_\to)$$

$$\frac{\Gamma \vdash S'_1 : \Delta_1, \ldots,\ S'_n : \Delta_n;\ U : (A \to B) \quad \Gamma \vdash S''_1 : \Delta_1, \ldots,\ S''_n : \Delta_n; V : A}{\Gamma \vdash S'_1 \cup S''_1 : \Delta_1, \ldots,\ S'_n \cup S''_n : \Delta_n; U \cup V : B}(E_\to)$$

$$\frac{\Gamma \vdash \Delta; S : A \wedge B}{\Gamma \vdash \Delta; S : A}(E^1_\wedge) \qquad \frac{\Gamma \vdash \Delta; S : A \wedge B}{\Gamma \vdash \Delta; S : B}(E^2_\wedge)$$

$$\frac{\Gamma \vdash S'_1 : \Delta_1, \ldots,\ S'_n : \Delta_n; U : A \quad \Gamma \vdash S''_1 : \Delta_1, \ldots,\ S''_n : \Delta_n;\ V : B}{\Gamma \vdash S'_1 \cup S''_1 : \Delta_1, \ldots,\ S'_n \cup S''_n : \Delta_n; U \cup V : A \wedge B}$$

$$\frac{\Gamma \vdash \Delta; S : A}{\Gamma \vdash \Delta; S : A \vee B}(I^1_\vee) \qquad \frac{\Gamma \vdash \Delta; S : B}{\Gamma \vdash \Delta; S : A \vee B}(I^2_\vee)$$

$$\frac{\Gamma \vdash \ldots, S_i : \Delta_i \ldots; S : A \vee B \quad \Gamma, A^x \vdash \ldots, S'_i : \Delta_i \ldots; S' : C \quad \Gamma, B^y \vdash \ldots, S''_i : \Delta_i \ldots; S'' : C}{\Gamma \vdash \ldots, S_i \cup S'_i[S/x] \cup S''_i[S/y] : \Delta_i \ldots; S'[S/x] \cup S''[S/y] : C}(E_\vee)$$

$$\frac{\Gamma \vdash S_1 : \Delta_1, \ldots, S_n : \Delta_n(, S_{n+1} : C); S : A \quad \Gamma, B^y \vdash S'_1 : \Delta_1, \ldots,\ S'_n : \Delta_n(, S_{n+1} : C); S' : C}{\Gamma \vdash S_1 \cup S'_1[S/y] : \Delta_1, \ldots,\ S_n \cup S'_n[S/y] : \Delta_n, S_{n+1} \cup S'_{n+1} \cup S'[S/y] : C; S : A - B}(I_-)$$

$$\frac{\Gamma \vdash S_1 : \Delta_1, \ldots, S_n : \Delta_n; S : A - B \quad \Gamma, A^x \vdash S'_1 : \Delta_1, \ldots,\ S'_n : \Delta_n; U : B}{\Gamma \vdash S_1 \cup S'_1[S/x] : \Delta_1, \ldots,\ S_n \cup S'_n[S/x] : \Delta_n; \{\} : C}(E_-)$$

FIGURE 3. Annotations for $\text{CND}_{\to\vee\wedge-}$

- derive right-hand side introduction/elimination rules for disjunction and subtraction (using the cut rule).

We shall just give two examples below, the full system of annotations for $\text{CND}_{\to\vee\wedge-}$ is summarized in Figure 3. Due to lack of space, we shorten $S_1 : \Delta_1, \ldots\ S_n : \Delta_n$ (with $n \geqslant 0$) simply as $\ldots, S_i : \Delta_i \ldots$ in the elimination rule for disjunction.

EXAMPLE 3.11
Let us derive the annotations for $(E_\vee)$. We first annotate the additive variant of the left-hand side introduction rule for disjunction (where hypotheses are still annotated):

$$\frac{T'_1 : \Gamma_1, \ldots, T'_p : \Gamma_p,\ U : A \vdash \Delta \quad T''_1 : \Gamma_1, \ldots, T''_p : \Gamma_p,\ V : B \vdash \Delta}{T'_1 \cup T''_1 : \Gamma_1, \ldots,\ T'_p \cup T''_p : \Gamma_p,\ U \cup V : A \vee B \vdash \Delta}$$

Let us now annotate this rule on the right-hand side:

$$\frac{\Gamma, A^x \vdash S'_1 : \Delta_1, \ldots,\ S'_n : \Delta_n \quad \Gamma, B^y \vdash S''_1 : \Delta_1, \ldots,\ S''_n : \Delta_n}{\Gamma, A \vee B^z \vdash S'_1[\{z\}/x] \cup S''_1[\{z\}/y] : \Delta_1, \ldots, S'_n[\{z\}/x] \cup S''_n[\{z\}/y] : \Delta_n}$$

Using the (additive variant of the) cut rule, we eventually obtain the right-hand side annotations for the usual (in natural deduction style) elimination rule for disjunction.

$$\frac{\Gamma \vdash \ldots, S_i : \Delta_i \ldots, S : A \vee B \quad \Gamma, A^x \vdash \ldots, S'_i : \Delta_i \ldots, S' : C \quad \Gamma, B^y \vdash \ldots, S''_i : \Delta_i \ldots, S'' : C}{\Gamma \vdash \ldots, S_i \cup S'_i[S/x] \cup S''_i[S/y] : \Delta_i \ldots,\ S'[S/x] \cup S''[S/y] : C}(E_\vee)$$

Note that the apparent inhomogeneous treatment of Δ_i and C comes from the fact that C cannot occur (with the same name) in the leftmost sequent of the premise.

EXAMPLE 3.12
Let us derive the annotations for (E_-). *We first annotate the left-hand side introduction rule for subtraction on the right-hand side:*

$$\frac{\Gamma, A^x \vdash S'_1 : \Delta_1, \ldots,\ S'_n : \Delta_n, U : B}{\Gamma, A - B^z \vdash S'_1[\{z\}/x] : \Delta_1, \ldots,\ S'_n[\{z\}/x] : \Delta_n}(\mathrm{LI}_-)$$

By applying the cut rule and then the weakening rule, we obtain the right-hand side annotations for the right-hand side elimination rule for subtraction:

$$\frac{\Gamma \vdash S_1 : \Delta_1, \ldots, S_n : \Delta_n; S : A - B \quad \dfrac{\Gamma, A^x \vdash S'_1 : \Delta_1, \ldots,\ S'_n : \Delta_n, U : B}{\Gamma, A - B^z \vdash S'_1[\{z\}/x] : \Delta_1, \ldots,\ S'_n[\{z\}/x] : \Delta_n}(\mathrm{LI}_-)}{\dfrac{\Gamma \vdash S_1 \cup S'_1[S/x] : \Delta_1, \ldots,\ S_n \cup S'_n[S/x] : \Delta_n}{\Gamma \vdash S_1 \cup S'_1[S/x] : \Delta_1, \ldots,\ S_n \cup S'_n[S/x] : \Delta_n, \{\} : C}(W_R)}(\mathrm{cut})$$

The technique used in the previous examples can be applied to show that the rules of Symmetric $\mathrm{CND}_{\to\vee\wedge-}$ and the rules of $\mathrm{CND}_{\to\vee\wedge-}$ are interderivable. Consequently, Symmetric $\mathrm{CND}_{\to\vee\wedge-}$ and $\mathrm{CND}_{\to\vee\wedge-}$ are equivalent as systems with annotations:

PROPOSITION 3.13
An annotated sequent is derivable in Symmetric $\mathrm{CND}_{\to\vee\wedge-}$ iff it is derivable in $\mathrm{CND}_{\to\vee\wedge-}$ (with the same annotations in both systems).

3.2.1 Restriction of $\mathrm{CND}_{\to\vee\wedge-}$ to subtractive logic

In order to define the system $\mathrm{SND}_{\to\vee\wedge-}$ we still have to derive the restriction on (E_-) from the restriction on (LI_-). We obtain thus:

DEFINITION 3.14
In an annotated proof of $\mathrm{CND}_{\to\vee\wedge-}$, we say that an occurrence of the elimination rule for subtraction is *constructive* iff $U \subseteq \{x\}$. The rule becomes then:

$$\frac{\Gamma \vdash S_1 : \Delta_1, \ldots, S_n : \Delta_n; S : A - B \quad \Gamma, A^x \vdash S'_1 : \Delta_1, \ldots,\ S'_n : \Delta_n; U : B}{\Gamma \vdash S_1 \cup S'_1[S/x] : \Delta_1, \ldots,\ S_n \cup S'_n[S/x] : \Delta_n; \{\} : C}(E_-) \quad \text{where } U \subseteq \{x\}$$

This definition extends to proofs:

DEFINITION 3.15
A proof of $\mathrm{CND}_{\to\vee\wedge-}$ is said to be *constructive* iff:

1. any occurrence of the introduction rule for implication is *constructive*,
2. any occurrence of the elimination rule for subtraction is *constructive*.

DEFINITION 3.16
We call $\mathrm{SND}_{\to\vee\wedge-}$ the restriction of $\mathrm{CND}_{\to\vee\wedge-}$ to constructive proofs.

By construction, constructive (E_{-}) and constructive (LI_{-}) are interderivable (with the same annotations), which gives us the following proposition:

PROPOSITION 3.17
An annotated sequent is derivable in Symmetric SND$_{\to\vee\wedge-}$ iff it is derivable in SND$_{\to\vee\wedge-}$ (with the same annotations in both systems).

THEOREM 3.18
SND$_{\to\vee\wedge-}$ is sound and complete for Subtractive Logic.

PROOF. By Proposition 3.17 and Theorem 3.10. ∎

Since subtractive logic is conservative over intuitionistic logic (see [8] for instance), we also have the following corollary:

COROLLARY 3.19
A sequent is derivable in SND $_{\to\vee\wedge}$ iff it is valid in intuitionistic logic.

EXAMPLE 3.20
In the following example the (only) occurrence of $(I_{\to})$ is constructive since there is no link between C and A in the premiss, and the derived sequent is thus valid in intuitionistic logic.

$$\dfrac{\dfrac{\dfrac{A \vee B^z \vdash ; A \vee B \qquad \dfrac{A^x \vdash ; A}{A^x \vdash \{x\} : A; \{\}B}(W_R) \qquad B^y \vdash ; B}{A \vee B^z \vdash \{z\} : A; \{z\} : B}(E_\vee) \qquad C^w \vdash ; \{w\} : C}{A \vee B^z, C^w \vdash \{z\}A; \{z, w\} : B \wedge C}(I_\wedge)}{A \vee B^z \vdash \{z\} : A; \{z\} : C \to (B \wedge C)}(I_\to)$$

4 The typed $\lambda\mu^{\to+\times-}$-calculus

In this section, we briefly present the $\lambda\mu^{\to+\times}$-calculus which is an extension of Parigot's $\lambda\mu$-calculus [36] where disjunction and conjunction are taken as primitives. Our $\lambda\mu^{\to+\times}$-calculus corresponds to de Groote's $\lambda\mu^{\to\wedge\vee\bot}$-calculus [12] up to minor differences (namely, we augment the syntax with a primitive **let** and we also consider simplification rules as in Parigot's original calculus). Then we extend this calculus with a primitive subtraction and thus obtain the $\lambda\mu^{\to+\times-}$-calculus. Note that in this 'unsafe' calculus coroutines are exactly as expressive as first-class continuations (since a coroutine's context is just a pair $\langle$*value*, *continuation*$\rangle$).

4.1 The $\lambda\mu^{\to+\times}$-calculus

Raw $\lambda\mu^{\to+\times}$-terms $M, N, \ldots$ are constructed by the following grammar:

$$\begin{array}{rcl} M & ::= & x \mid (M\ N) \mid \lambda x.M \mid \textbf{let}\ x = M\ \textbf{in}\ N \mid \mu\alpha M \mid [\beta]M \\ & \mid & \textbf{inl}\ M \mid \textbf{inr}\ M \mid \textbf{case}\ M\ \textbf{of}\ (\textbf{inl}\ x) \mapsto P \mid (\textbf{inr}\ y) \mapsto Q \\ & \mid & \langle M, N\rangle \mid \textbf{fst}\ M \mid \textbf{snd}\ M \end{array}$$

where $x, y, z, \ldots$ range over variables and $\alpha, \beta, \gamma, \ldots$ range over names. Besides, we call *simple $\lambda\mu^{\to+\times}$-contexts* the contexts defined by the following grammar:

$$C \quad ::= \quad ([\,]\ t) \mid \textbf{fst}\ [\,] \mid \textbf{snd}\ [\,] \mid \textbf{case}\ [\,]\ \textbf{of}\ (\textbf{inl}\ x) \mapsto M \mid (\textbf{inr}\ \ y) \mapsto N$$

4.1.1 Reduction rules

Detour-reduction

(a) $(\lambda x.u\ t) \quad \rightsquigarrow \quad u\{t/x\}$

(b) **fst** $\langle t, u\rangle \quad \rightsquigarrow \quad t$

(c) **snd** $\langle t, u\rangle \quad \rightsquigarrow \quad u$

(d) **case** (**inl** t) **of** (**inl** x) $\mapsto u \mid$ (**inr** y) $\mapsto v \quad \rightsquigarrow \quad u\{t/x\}$

(e) **case** (**inr** t) **of** (**inl** x) $\mapsto u \mid$ (**inr** y) $\mapsto v \quad \rightsquigarrow \quad v\{t/y\}$

(f) **let** $x = t$ **in** $u \quad \rightsquigarrow \quad u\{t/x\}$

where $u\{t/x\}$ stands for the usual capture-avoiding substitution.

Structural reduction

(μ-reduction and permutative rule)

(a) $C[\mu\alpha.u] \quad \rightsquigarrow \quad \mu\alpha.u\{\alpha \leftarrow C[\,]\}$

(b) $C[$**case** t **of** (**inl** x) $\mapsto u \mid$ (**inr** y) $\mapsto v] \quad \rightsquigarrow \quad$ **case** t **of** (**inl** x) $\mapsto C[u] \mid$ (**inr** y) $\mapsto C[v]$

where $C[\,]$ ranges over simple $\lambda\mu^{\rightarrow+\times}$-contexts and $M\{\alpha \leftarrow C[\,]\}$ denotes the structural substitution which is inductively defined as follows, where M^* stands for $M\{\alpha \leftarrow C[\,]\}$:

$$
\begin{array}{rcl}
x^* & \equiv & x \\
(\lambda x.t)^* & \equiv & \lambda x.t^* \\
(t\ u)^* & \equiv & (t^*\ u^*) \\
(\textbf{fst}\ t)^* & \equiv & \textbf{fst}\ t^* \text{ and } (\textbf{snd}\ t)^* \ \equiv\ \textbf{snd}\ t^* \\
\langle t, u\rangle^* & \equiv & \langle t^*, u^*\rangle \\
(\textbf{case}\ t\ \textbf{of}\ (\textbf{inl}x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v)^* & \equiv & \textbf{case}\ t^*\ \textbf{of}\ (\textbf{inl}x) \mapsto u^* \mid (\textbf{inr}\ y) \mapsto v^* \\
(\textbf{inl}\ t)^* & \equiv & \textbf{inl}\ t^* \text{ and } (\textbf{inr}\ t)^* \ \equiv\ \textbf{inr}\ t^* \\
(\textbf{let}\ x = t\ \textbf{in}\ u)^* & \equiv & \textbf{let}\ x = t^*\ \textbf{in}\ u^* \\
(\mu\gamma.u)^* & \equiv & \mu\gamma.u^* \\
([\alpha]t)^* & \equiv & [\alpha]C[t^*] \\
([\gamma]t)^* & \equiv & [\gamma]t^* \text{ if } \gamma \neq \alpha
\end{array}
$$

Simplification

(a) $\mu\alpha[\alpha]u \quad \rightsquigarrow \quad u$ if α does not occur in u

(b) $[\beta]\mu\alpha.t \quad \rightsquigarrow \quad t\{\beta/\alpha\}$

REMARK 4.1

We shall often refer to the following macro-definitions: **get-context** $\alpha\ t \equiv \mu\alpha[\alpha]t$ and **set-context** $\alpha\ t \equiv \mu\delta[\alpha]t$ where δ is a name which does not occur in t (see [7] for a study of the sublanguage obtained when we restrict the $\lambda\mu$-terms to these operators, where they are called respectively **catch** and **throw**). Our terminology in this paper comes from [24].

$$x : A^x \vdash ; A$$

$$\frac{t : \Gamma \vdash \Delta; B}{t : \Gamma, A^x \vdash \Delta; B}(W_L) \qquad \frac{t : \Gamma \vdash \Delta; B}{t : \Gamma \vdash \Delta, A^\alpha \ ; B}(W_R)$$

$$\frac{t : \Gamma \vdash \Delta(, A^\alpha);}{\mu\alpha.t : \Gamma \vdash \Delta; A}(\textit{activate}) \qquad \frac{t : \Gamma \vdash \Delta(, A^\alpha); A}{[\alpha]t : \Gamma \vdash \Delta, A^\alpha;}(\textit{passivate})$$

$$\frac{t : \Gamma, A^x \vdash \Delta; B}{\lambda x.t : \Gamma \vdash \Delta; A \to B}(I_\to) \qquad \frac{u : \Gamma \vdash \Delta; A \to B \quad v : \Gamma \vdash \Delta; A}{(u\ v) : \Gamma \vdash \Delta; B}(E_\to)$$

$$\frac{t : \Gamma \vdash \Delta; A}{\textbf{inl}\ t : \Gamma \vdash \Delta; A \vee B}(I^1_\vee) \qquad \frac{t : \Gamma \vdash \Delta; B}{\textbf{inr}\ t : \Gamma \vdash \Delta; A \vee B}(I^2_\vee)$$

$$\frac{t : \Gamma \vdash \Delta; A \vee B \quad u : \Gamma, A^x \vdash \Delta; C \quad v : \Gamma, B^y \vdash \Delta; C}{\textbf{case}\ t\ \textbf{of}\ (\textbf{inl}\ x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v : \Gamma \vdash \Delta; C}(E_\vee)$$

$$\frac{t : \Gamma \vdash \Delta; A \wedge B}{\textbf{fst}\ t : \Gamma \vdash \Delta; A}(E^1_\wedge) \qquad \frac{t : \Gamma \vdash \Delta; A \wedge B}{\textbf{snd}\ t : \Gamma \vdash \Delta; B}(E^2_\wedge)$$

$$\frac{t : \Gamma \vdash \Delta; A \quad u : \Gamma \vdash \Delta; B}{\langle t, u\rangle : \Gamma \vdash \Delta; A \wedge B}(I_\wedge)$$

$$\frac{u : \Gamma \vdash \Delta; A \quad t : \Gamma, A^x \vdash \Delta; B}{\textbf{let}\ x = u\ \textbf{in}\ t : \Gamma \vdash \Delta; B}(\textit{cut})$$

FIGURE 4. The $\lambda\mu^{\to+\times}$-calculus

4.2 Typing rules for the $\lambda\mu^{\to+\times}$-calculus

The typing rules for the $\lambda\mu^{\to+\times}$-calculus are summarized in Figure 4. Note that the typing rules for **get-context** and **set-context** are easily derivable. They correspond respectively to the right-hand side weakening and contraction rules :

$$\frac{t : \Gamma \vdash \Delta; A}{\textbf{set-context}\ \alpha\ t : \Gamma \vdash \Delta, A^\alpha; B}(W_R) \qquad \frac{t : \Gamma \vdash \Delta, A^\alpha; A}{\textbf{get-context}\ \alpha\ t : \Gamma \vdash \Delta; A}(C_R)$$

REMARK 4.2
In order to deal with the negation, we add a macro-definition called **abort** (following de Groote [10]) defined as **set-context** $\varepsilon\ t$ where ε is a free name (the name of $\bot$ given in section 2.1) considered as a constant (see [1] for more details). Here is the typing rule for **abort**:

$$\frac{t : \Gamma \vdash \Delta; \bot}{\textbf{abort}\ t : \Gamma \vdash \Delta; C}(\bot_E)$$

4.3 The $\lambda\mu^{\to+\times-}$-calculus

4.3.1 Defining $A - B$ as $A \wedge \neg B$

We have shown in Section 2.1 that the introduction/elimination rules for subtraction are derivable if we define $A - B$ as $A \wedge \neg B$. Since we are looking for a term calculus for CND$_{\to\vee\wedge-}$, let us first

annotate these proofs with $\lambda\mu^{\rightarrow+\times}$-terms and consider the macro-definitions we obtain.

Derivation of the introduction rule

$$\dfrac{t : \Gamma \vdash \Delta; A \qquad \dfrac{\dfrac{x : \Gamma, B^x \vdash \Delta; B}{\textbf{set-context}\ \beta\ x : \Gamma, B^x \vdash \Delta, B^\beta; \bot}(\textit{set})}{\lambda x.\textbf{set-context}\ \beta\ x : \Gamma \vdash \Delta, B^\beta; \neg B}(I_\rightarrow)}{(t, \lambda x.\textbf{set-context}\ \beta\ x) : \Gamma \vdash \Delta, B^\beta; A \wedge \neg B}(I_\wedge)$$

Derivation of the elimination rule

$$\dfrac{t : \Gamma \vdash \Delta; A \wedge \neg B \qquad \dfrac{\dfrac{k : \neg B^k \vdash \neg B \quad u : \Gamma, A^x \vdash \Delta; B}{(k\ u) : \Gamma, A^x, \neg B^k \vdash \Delta; \bot}(E_\rightarrow)}{\textbf{abort}\ (k\ u) : \Gamma, A^x, \neg B^k \vdash \Delta; C}(E_\bot)}{\textbf{match}\ t\ \textbf{with}\ (x, k) \mapsto \textbf{abort}\ (k\ u) : \Gamma \vdash \Delta; C}(E_\wedge)$$

REMARK 4.3

- It is easy to define the **match** instruction used in the last inference rule with the projections, for instance as follows:

  $$\textbf{match}\ t\ \textbf{with}\ (x, y) \mapsto u \equiv \textbf{let}\ x = (\textbf{fst}\ t)\ \textbf{in}\ (\textbf{let}\ y = (\textbf{snd}\ t)\ \textbf{in}\ u).$$

  However, since we also expect the subformula property for the $\lambda\mu^{\rightarrow+\times-}$-calculus, we need permutative reduction rules related to **match** (in order to derive the permutative reduction rules related to **resume**) and these rules are usually not derivable when pattern matching is defined using the projections. This is why we need to consider an extension of the $\lambda\mu^{\rightarrow+\times\bot}$-calculus with a new product $\otimes$ and built-in pattern matching. The resulting calculus (called the $\lambda\mu^{\rightarrow+\times\otimes\bot}$-calculus) is studied in appendix A.
- Informally, the introduction rules builds a pair (*value*, *continuation*) while the elimination rule opens a pair (*value*, *continuation*) and invokes the *continuation* on some $\lambda\mu$-term which may also use the *value*. We shall explain in Section 5.8 why we interpret such a pair as a first-class coroutine, and the *value* as its local environment. For the time being, we just define the following macros:

  $$\begin{array}{rcl} \textbf{make-coroutine}\ t\ \alpha & \equiv & (t, \lambda z.\textbf{set-context}\ \alpha\ z) \\ \textbf{resume}\ t\ \textbf{with}\ x \mapsto u & \equiv & \textbf{match}\ t\ \textbf{with}\ (x, k) \mapsto\ \textbf{abort}\ (k\ u). \end{array}$$

  The derived typing rules for these macro-definitions are then:

  $$\dfrac{t : \Gamma \vdash \Delta; A}{\textbf{make-coroutine}\ t\ \beta : \Gamma \vdash \Delta, B^\beta; A - B} \qquad \dfrac{t : \Gamma \vdash \Delta; A - B \quad u : \Gamma, A^x \vdash \Delta; B}{\textbf{resume}\ t\ \textbf{with}\ x \mapsto u : \Gamma \vdash \Delta; C}$$

- We said in the introduction that in the $\lambda\mu$-calculus, a name α may be reified as the first-class continuation $\lambda z.\textbf{set-context}\ \alpha\ z$. However, this last syntactic form is not closed under μ-reduction. For instance, the term $((\textbf{get-context}\ \alpha\ (\lambda z.\textbf{set-context}\ \alpha\ z))\ u)$ is a μ-redex, and its contractum is $(\textbf{get-context}\ \alpha\ ((\lambda z.\textbf{set-context}\ \alpha\ (z\ u))\ u))$. We shall thus consider a more general form $\lambda z.\textbf{set-context}\ \alpha\ C[z]$ where $C[\,]$ is a continuation context (see Definition 4.4).

We are now ready to extend the $\lambda\mu^{\rightarrow+\times}$-calculus with first-class coroutines.

4.3.2 Syntax of the $\lambda\mu^{\rightarrow+\times-}$-calculus

We add two term constructors to the syntax of the raw $\lambda\mu^{\rightarrow+\times}$-calculus (where $C[\,]$ ranges over arbitrary contexts):

$$M \quad ::= \quad \ldots \mid \textbf{make-coroutine}\ M\ (C[\,], \alpha) \mid \textbf{resume}\ M\ \textbf{with}\ x \mapsto N.$$

We also define the *simple* $\lambda\mu^{\rightarrow+\times-}$*-contexts* by extending the grammar of simple $\lambda\mu^{\rightarrow+\times}$-contexts:

$$C \quad ::= \quad \ldots \mid \textbf{resume}\ [\,]\ \textbf{with}\ x \mapsto N.$$

4.3.3 Continuation contexts

DEFINITION 4.4

Continuation contexts are defined by the following grammar:

$$\begin{array}{rcl} C & ::= & [\,] \quad \mid \quad (C[\,]\ t) \\ & \mid & \textbf{fst}\ C[\,] \quad \mid \quad \textbf{snd}\ C[\,] \\ & \mid & \textbf{case}\ C[\,]\ \textbf{of}\ (\textbf{inl}\ x) \mapsto M \mid (\textbf{inr}\ \ y) \mapsto N \\ & \mid & \textbf{resume}\ C[\,]\ \textbf{with}\ x \mapsto N. \end{array}$$

REMARK 4.5

We shall abbreviate as C_α a pair $(C[\,], \alpha)$ where $C[\,]$ is a continuation context. Note that if $C[\,]$ is a simple $\lambda\mu^{\rightarrow+\times-}$-context and $K[\,]$ is a continuation context then $C[K[\,]]$ is also a continuation context, and thus $C[K]_\alpha$ is an abbreviation for $(C[K[\,]], \alpha)$.

4.3.4 Reduction rules

We extend the reduction rules of the $\lambda\mu^{\rightarrow+\times}$-calculus with this new detour-reduction rule:

(g) $\textbf{resume}\ (\textbf{make-coroutine}\ t\ C_\alpha)\ \textbf{with}\ x \mapsto u \rightsquigarrow \textbf{set-context}\ \alpha\ C[u\{t/x\}]$

and the structural reduction is now defined by the following rules:

(a) $C[\mu\alpha.u] \quad \rightsquigarrow \quad \mu\alpha.u\{\alpha \hookleftarrow C[\,]\}$

(b) $C[\textbf{case}\ t\ \textbf{of}\ (\textbf{inl}\ x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v] \quad \rightsquigarrow \quad \textbf{case}\ t\ \textbf{of}\ (\textbf{inl}\ x) \mapsto C[u] \mid (\textbf{inr}\ y) \mapsto C[v]$

(c) $C[\textbf{resume}\ t\ \textbf{with}\ x \mapsto u] \quad \rightsquigarrow \quad \textbf{resume}\ t\ \textbf{with}\ x \mapsto u$

where $C[\,]$ ranges over simple $\lambda\mu^{\rightarrow+\times-}$-contexts. We also extend the inductive definition of the structural substitution $t\{\alpha \hookleftarrow C[\,]\}$ to $\lambda\mu^{\rightarrow+\times-}$-terms and continuation contexts as follows, where again M^* stands for $M\{\alpha \hookleftarrow C[\,]\}$:

$$\begin{array}{rcl} [\,]^* & \equiv & [\,] \\ (\textbf{make-coroutine}\ t\ K_\alpha)^* & \equiv & \textbf{make-coroutine}\ t^*\ C[K^*]_\alpha \\ (\textbf{make-coroutine}\ t\ K_\gamma)^* & \equiv & \textbf{make-coroutine}\ t^*\ K_\gamma^* \text{ if } \gamma \neq \alpha \\ (\textbf{resume}\ t\ \textbf{with}\ x \mapsto u)^* & \equiv & \textbf{resume}\ t^*\ \textbf{with}\ x \mapsto u^*. \end{array}$$

REMARK 4.6

By definition of the structural substitution, the form $\textbf{make-coroutine}\ t\ C_\alpha$ where $C[\,]$ is a continuation context is preserved by the reduction rules of the $\lambda\mu^{\rightarrow+\times-}$-calculus. We shall thus restrict the set of raw $\lambda\mu^{\rightarrow+\times-}$-terms to those terms.

4.4 Typing rules for coroutines

The typing rules for **make-coroutine** and **resume** are respectively introduction and elimination rules for subtraction:

$$\frac{t : \Gamma \vdash \Delta; A \quad C[x] : \Gamma, D^x \vdash \Delta; B}{\textbf{make-coroutine}\ t\ C_\beta : \Gamma \vdash \Delta, B^\beta; A - D}(I_-) \qquad \frac{t : \Gamma \vdash \Delta; A - B \quad u : \Gamma, A^x \vdash \Delta; B}{\textbf{resume}\ t\ \textbf{with}\ x \mapsto u : \Gamma \vdash \Delta; C}(E_-)$$

REMARK 4.7
The typing rule for **make-coroutine** given in the Remark 4.3 corresponds to the particular case $C[\,] = [\,]$ (but we have proved in Section 2 that from a logical standpoint these rules are equivalent).

4.5 Strong normalization and the Church–Rosser property

THEOREM 4.8
The typed $\lambda\mu^{\rightarrow+\times-}$-calculus is strongly normalizing and enjoys the Church–Rosser property.

PROOF. (sketch) Recall that the typed $\lambda\mu^{\rightarrow+\times\otimes\bot}$-calculus is an extension of the $\lambda\mu^{\rightarrow+\times\bot}$-calculus with pattern matching (for the tensor product$\otimes$) which is strongly normalizing and enjoys the Church–Rosser property (see Appendix A). Let us denote by Φ the translation from the $\lambda\mu^{\rightarrow+\times-}$-calculus into the $\lambda\mu^{\rightarrow+\times\otimes\bot}$-calculus defined by the following macro-definitions:

$$\begin{array}{rcl} \textbf{make-coroutine}\ t\ C_\alpha & \equiv & (t, \lambda z.\textbf{set-context}\ \alpha\ C[z]) \\ \textbf{resume}\ t\ \textbf{with}\ x \mapsto u & \equiv & \textbf{match}\ t\ \textbf{with}\ (x,k) \mapsto\ \textbf{abort}\ (k\ u). \end{array}$$

It is sufficient to check the following properties:

1. Φ is a morphism for the reduction: if $u \rightsquigarrow v$ then $\Phi(u) \rightsquigarrow^+ \Phi(v)$.
2. Φ preserves normal forms: if u is a normal $\lambda\mu^{\rightarrow+\times-}$-term then $\Phi(u)$ is a normal $\lambda\mu^{\rightarrow+\times\otimes\bot}$-term.
3. Φ is injective on normal forms.

Indeed, from the first property, we can derive the strong normalization of the $\lambda\mu^{\rightarrow+\times-}$-calculus from the strong normalization of the $\lambda\mu^{\rightarrow+\times\otimes\bot}$-calculus. From the second and third properties, we obtain the uniqueness of the normal form (i.e. the Church–Rosser property). The proof that Φ is a morphism for the reduction is given in Appendix A. ∎

4.6 Normal forms and the subformula property

We end this section by proving the subformula property (which is derived as usual from a characterization of the normal terms).

PROPOSITION 4.9
Given the derivation, in system CND$_{\rightarrow\vee\wedge-}$, of some typing judgement $t\ : \Gamma \vdash\ \Delta; A$, if t is normal then every type occurring in the derivation is either a subformula of a type occurring in Γ or Δ, or a subformula of A.

PROOF. Since the subformula property always holds for introduction rules (and weakening/ contraction rules), we just have to check the property for occurrences of elimination rules in a normal proof. Let us then call *applicative contexts* the contexts defined by the following grammar:

$$A \quad ::= \quad [\,] \quad | \quad (A\ t) \quad | \quad \textbf{fst}\ A \quad | \quad \textbf{snd}\ A$$

In a well-typed normal $\lambda\mu^{\rightarrow+\times-}$-term, any occurrence of an application or a projection has the form $A[x]$ (where x is a variable). Indeed, in an application $(u\ v)$ and in a projection **fst** u or **snd** u, the term u must be either a variable, or again a projection or an application (otherwise we obtain a detour-redex or a redex for one of the structural rules). It is easy to prove by induction that in the typing judgement of an applicative context $A[x] : \Gamma, F^x \vdash \Delta; D$, the formula D is a subformula of F.

Now, in a well-typed normal $\lambda\mu^{\rightarrow+\times-}$-term, in any subterm of the form **case** t **of** $(\textbf{inl}\ x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v$ or **resume** t **with** $x \mapsto u$, the term t has the form $A[z]$ (where z is a variable). Indeed, otherwise we obtain a detour-redex or redex for one of the structural rules. Consequently, any occurrence of an elimination rule for $\vee$ (resp. $-$) has the following form:

Elimination rule for $\vee$

$$\frac{A[z] : \Gamma, F^z \vdash \Delta; B \vee C \quad u : \Gamma, B^x \vdash \Delta; D \quad v : \Gamma; C^y \vdash \Delta; D}{\textbf{case}\ A[z]\ \textbf{of}\ \ (\textbf{inl}\ x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v : \Gamma, F^z \vdash \Delta; D}$$

Elimination rule for $-$

$$\frac{A[z] : \Gamma, F^z \vdash \Delta; A - B \quad u : \Gamma, A^x \vdash \Delta; B}{\textbf{resume}\ A[z]\ \textbf{with}\ x \mapsto u : \Gamma, F^z \vdash \Delta; C}$$

Then one can easily check that the subformula property holds for these rules. ∎

5 The safe $\lambda\mu$-calculus

From a computational standpoint, the restriction of the introduction rule for $\rightarrow$ to at most one conclusion amounts to requiring that in any subterm $\lambda x.u$ of a term t, u is μ-closed (there is no free name in u). This restriction is clearly not closed under reduction. For instance, if u contains a free name (take for instance $u = \mu\beta[\alpha]\lambda z.z$):

$$((\lambda y \lambda x.y)\ u) \rightsquigarrow \lambda x.u$$

then in the contractum u is not μ-closed. We define in this section a weaker restriction on $\lambda\mu^{\rightarrow+\times}$-terms which is closed under reduction. This restriction has actually been derived from the restriction on sequents given in Section 3.1.2. The formal relation between the two definitions is stated by Theorem 5.9.

5.1 The safe $\lambda\mu^{\rightarrow+\times-}$-calculus

In this section we introduce the concept of 'scope of a name with respect to variables' which corresponds to the variables shared by a coroutine. This leads to the formal definition of a $\lambda\mu^{\rightarrow+\times}$-term in which a coroutine does not access the local environment of another coroutine (see Remark 5.8). We shall say that such a $\lambda\mu^{\rightarrow+\times-}$-term is *safe with respect to coroutine contexts*. Then we give the formal definition of a $\lambda\mu^{\rightarrow+\times-}$-term *safe with respect to first-class coroutines*.

We call $\mathcal{S}_{[]}(t)$ the set of free variables that occur in the scope of the current coroutine (which we dubbed $[]$) and $\mathcal{S}_\delta(t)$ the set of the variables that occur in the scope of a coroutine δ in t. It is easy to check by induction on t that $\mathcal{S}_\delta(t) = \emptyset$ if δ does not occur free in t.

DEFINITION 5.1
For any $\lambda\mu^{\rightarrow+\times-}$-term t, the inductive definition of the sets $\mathcal{S}_{[]}(t)$ and $\mathcal{S}_\delta(t)$ is given in Figure 5.

- $\mathcal{S}_{[]}(x) = \{x\}$
  $\mathcal{S}_{\delta}(x) = \emptyset$
- $\mathcal{S}_{[]}(\lambda x.u) = \mathcal{S}_{[]}(u)\backslash\{x\}$
  $\mathcal{S}_{\delta}(\lambda x.u) = \mathcal{S}_{\delta}(u)\backslash\{x\}$
- $\mathcal{S}_{[]}(u\ v) = \mathcal{S}_{[]}(u) \cup \mathcal{S}_{[]}(v)$
  $\mathcal{S}_{\delta}(u\ v) = \mathcal{S}_{\delta}(u) \cup \mathcal{S}_{\delta}(v)$
- $\mathcal{S}_{[]}([\alpha]u) = \emptyset$
  $\mathcal{S}_{\delta}([\alpha]u) = \mathcal{S}_{\delta}(u)$ for any $\delta \neq \alpha$ and $\mathcal{S}_{\alpha}([\alpha]u) = \mathcal{S}_{\alpha}(u) \cup \mathcal{S}_{[]}(u)$
- $\mathcal{S}_{[]}(\mu\alpha.u) = \mathcal{S}_{\alpha}(u)$
  $\mathcal{S}_{\delta}(\mu\alpha.u) = \mathcal{S}_{\delta}(u)$
- $\mathcal{S}_{[]}(\textbf{inl}\ u) = \mathcal{S}_{[]}(u)$ and $\mathcal{S}_{[]}(\textbf{inr}\ u) = \mathcal{S}_{[]}(u)$
  $\mathcal{S}_{\delta}(\textbf{inl}\ u) = \mathcal{S}_{\delta}(u)$ and $\mathcal{S}_{\delta}(\textbf{inr}\ u) = \mathcal{S}_{\delta}(u)$
- $\mathcal{S}_{[]}(\textbf{case}\ w\ \textbf{of}\ (\textbf{inl}\ x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v) = \mathcal{S}_{[]}(u)[\mathcal{S}_{[]}(w)/x] \cup \mathcal{S}_{[]}(v)[\mathcal{S}_{[]}(w)/y]$
  $\mathcal{S}_{\delta}(\textbf{case}\ w\ \textbf{of}\ (\textbf{inl}\ x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v) = \mathcal{S}_{\delta}(u)[\mathcal{S}_{[]}(w)/x] \cup \mathcal{S}_{\delta}(v)[\mathcal{S}_{[]}(w)/y] \cup \mathcal{S}_{\delta}(w)$
- $\mathcal{S}_{[]}(\langle t, u\rangle) = \mathcal{S}_{[]}(t)\ \cup \mathcal{S}_{[]}(u)$
  $\mathcal{S}_{\delta}(\langle t, u\rangle) = \mathcal{S}_{\delta}(t)\ \cup \mathcal{S}_{\delta}(u)$
- $\mathcal{S}_{[]}(\textbf{fst}\ u) = \mathcal{S}_{[]}(u)$ and $\mathcal{S}_{[]}(\textbf{snd}\ u) = \mathcal{S}_{[]}(u)$
  $\mathcal{S}_{\delta}(\textbf{fst}\ u) = \mathcal{S}_{\delta}(u)$ and $\mathcal{S}_{\delta}(\textbf{snd}\ u) = \mathcal{S}_{\delta}(u)$
- $\mathcal{S}_{[]}(\textbf{let}\ x = v\ \textbf{in}\ u) = \mathcal{S}_{[]}(u)[\mathcal{S}_{[]}(v)/x]$
  $\mathcal{S}_{\delta}(\textbf{let}\ x = v\ \textbf{in}\ u) = \mathcal{S}_{\delta}(u)[\mathcal{S}_{[]}(v)/x] \cup \mathcal{S}_{\delta}(v)$
- $\mathcal{S}_{[]}([\,]) = \emptyset$
  $\mathcal{S}_{\delta}([\,]) = \emptyset$
- $\mathcal{S}_{[]}(\textbf{make-coroutine}\ t\ C_{\alpha}) = \mathcal{S}_{[]}(t)$
  $\mathcal{S}_{\alpha}(\textbf{make-coroutine}\ t\ C_{\alpha}) = \mathcal{S}_{\alpha}(t) \cup \mathcal{S}_{\alpha}(C[\,]) \cup \mathcal{S}_{[]}(t) \cup \mathcal{S}_{[]}(C[\,])$
  $\mathcal{S}_{\delta}(\textbf{make-coroutine}\ t\ C_{\alpha}) = \mathcal{S}_{\delta}(t) \cup \mathcal{S}_{\delta}(C[\,])$ for any $\delta \neq \alpha$
- $\mathcal{S}_{[]}(\textbf{resume}\ c\ \textbf{with}\ x \mapsto u) = \mathcal{S}_{[]}(u)\backslash\{x\} \cup \mathcal{S}_{[]}(c)$
  $\mathcal{S}_{\delta}(\textbf{resume}\ c\ \textbf{with}\ x \mapsto u) = \mathcal{S}_{\delta}(u)[\mathcal{S}_{[]}(c)/x] \cup \mathcal{S}_{\delta}(c)$

FIGURE 5. Shared variables

REMARK 5.2
In the particular case of **set-context** and **get-context**, we obtain the following definition:

- If the $\lambda\mu^{\rightarrow+\times-}$-term is **get-context** $\alpha\ u$ (*i.e.* $\mu\alpha[\alpha]u$) then:
  $\mathcal{S}_{[]}(\mu\alpha[\alpha]u) = \mathcal{S}_{\alpha}([\alpha]u) = \mathcal{S}_{[]}(u) \cup \mathcal{S}_{\alpha}(u)$
  $\mathcal{S}_{\delta}(\mu\alpha[\alpha]u) = \mathcal{S}_{\delta}([\alpha]u) = \mathcal{S}_{\delta}(u)$.
- If the $\lambda\mu^{\rightarrow+\times-}$-term t is **set-context** $\alpha\ u$ (*i.e.* $\mu\beta[\alpha]u$ where β does not occur in $[\alpha]u$) then:
  $\mathcal{S}_{[]}(\mu\beta[\alpha]u) = \mathcal{S}_{\beta}([\alpha]u) = \emptyset$
  $\mathcal{S}_{\delta}(\mu\beta[\alpha]u) = \mathcal{S}_{\delta}([\alpha]u) = \mathcal{S}_{\delta}(u)$ for any $\delta \neq \alpha$ and $\mathcal{S}_{\alpha}(\mu\beta[\alpha]u) = \mathcal{S}_{\alpha}(u) \cup \mathcal{S}_{[]}(u)$.

REMARK 5.3
Given a $\lambda\mu^{\rightarrow+\times-}$-term, a variable may occur in the scope of *several* coroutines (including the current coroutine). We shall say that such a variable is *shared* between several coroutines.

DEFINITION 5.4
A $\lambda\mu^{\rightarrow+\times-}$-term t is ***safe with respect to coroutine contexts*** iff for any subterm of t which has the form $\lambda x.u$, for any free name δ of u, $x \notin \mathcal{S}_\delta(u)$.

REMARK 5.5
In safe $\lambda\mu^{\rightarrow+\times}$-terms, the usual abbreviation $(\lambda x.t)\ u$ is no longer equivalent to **let** $x = u$ **in** t since in $(\lambda x.t)\ u$, the variable x may not occur in the scope of some name in t: this declaration of x is local to the current coroutine. On the other hand, the definition of x in **let** $x = u$ **in** t is global: any coroutine may access x in t.

Note that although global definitions are useful from a programming perspective, in this paper, the main purpose of **let** is just to avoid using the (meta-level) substitution in the term interpretation of the cut rule.

EXAMPLE 5.6
As expected, a first class continuation like $\lambda y.$**set-context** $\beta\ y$ is not safe with respect to coroutine contexts. On the other hand the following example is safe:

$$\lambda f.\textbf{get-context}\ \alpha\ \lambda x.\textbf{get-context}\ \beta\ \textbf{set-context}\ \alpha\ (f\ (\textbf{set-context}\ \beta\ x)).$$

Indeed, in context α the variable f is visible (even if x is not) while in context β they are both visible.

DEFINITION 5.7
A $\lambda\mu^{\rightarrow+\times-}$-term t is ***safe with respect to*** **first-class coroutines** (or just '*safe*' for short) iff:

1. for any subterm of t which has the form $\lambda x.u$, for any free name δ of u, $x \notin \mathcal{S}_\delta(u)$ (i.e. t is safe with respect to coroutine contexts).
2. for any subterm of t which has the form **resume** c **with** $x \mapsto u$, $\mathcal{S}_{[]}(u) \subseteq \{x\}$.

REMARK 5.8
If we violate one of these restrictions, we recover the (unsafe) $\lambda\mu^{\rightarrow+\times}$-calculus (and classical logic as a type system). For the first restriction, it is obvious. For the second restriction, consider the following $\lambda\mu^{\rightarrow+\times-}$-term:

$$\lambda y.(\textbf{resume}\ (\textbf{make-coroutine}\ e\ \beta)\ \textbf{with}\ x \mapsto y) \rightsquigarrow \lambda y.\textbf{set-context}\ \beta\ y.$$

The contractum is indeed a first-class continuation. When it is resumed, a first-class coroutine is allowed to access only its own local data (which is packed together with the continuation). This is why we claim that such a pair (*value*, *continuation*) corresponds actually (for safe $\lambda\mu^{\rightarrow+\times-}$-terms) to a pair (*environment*, *continuation*) where the *environment* contains the only data visible in the coroutine when it resumes.

5.2 *Typing safe* $\lambda\mu^{\rightarrow+\times-}$*-terms in* $\mathrm{SND}_{\rightarrow\vee\wedge-}$

Proofs of $\mathrm{SND}_{\rightarrow\vee\wedge-}$ (i.e. constructive proofs of $\mathrm{CND}_{\rightarrow\vee\wedge-}$) and safe $\lambda\mu^{\rightarrow+\times-}$-terms are related by this theorem (and its corollary):

THEOREM 5.9
Given an annotated derivation, in system $\mathrm{CND}_{\rightarrow\vee\wedge-}$, of some typing judgement $t\ :\ \Gamma_1^{x_1}, \ldots, \Gamma_n^{x_n} \vdash \Delta_1^{\alpha_1}, \ldots, \Delta_m^{\alpha_m}\ ; A$, then $x_i \in \mathcal{S}_{\alpha_j}(t)$ iff there is a link between $\Gamma_i^{x_i}$ and $\Delta_j^{\alpha_j}$ and $x_i \in \mathcal{S}_{[]}(t)$ iff there is a link between $\Gamma_i^{x_i}$ and A.

PROOF. We check that we have two inductive definitions of the same dependency relations between hypotheses and conclusions of a sequent (recall that $[]$ is the name of the formula which is on the right-hand side of the semi-colon). Take for instance the rule $(E_{\rightarrow})$ of CND$_{\rightarrow\vee\wedge-}$:

$$\frac{t : \Gamma \vdash S'_1 : \Delta_1^{\delta_1}, \ldots,\ S'_n : \Delta_n^{\delta_n};\ U : (A \rightarrow B) \quad u : \Gamma \vdash S''_1 : \Delta_1^{\delta_1}, \ldots,\ S''_n : \Delta_n^{\delta_n};\ V : A}{(t\ u) : \Gamma \vdash S'_1 \cup S''_1 : \Delta_1^{\delta_1}, \ldots,\ S'_n \cup S''_n : \Delta_n^{\delta_n}; U \cup V : B}$$

By induction hypothesis, $S'_i = S_{\delta_i}(t)$, $S''_i = S_{\delta_i}(u)$, $U = S_{[]}(t)$ and $V = S_{[]}(u)$ and by definition 5.1:

$$\begin{array}{rcl} \mathcal{S}_{[]}(t\ u) & = & \mathcal{S}_{[]}(t) \cup \mathcal{S}_{[]}(u) \ = \ U \cup V \\ \mathcal{S}_{\delta_i}(t\ u) & = & \mathcal{S}_{\delta_i}(t) \cup \mathcal{S}_{\delta_i}(u) \ = \ S'_i \cup S''_i. \end{array}$$

The only difficulty comes from the introduction rule of subtraction (I_-) but since $C[\,]$ is a continuation context, we know that y occurs in S' and nowhere else. We obtain thus:

$$\frac{t : \Gamma \vdash \ldots, S_i : \Delta_i^{\delta_i}, \ldots (, S_{n+1} : C^{\beta}); S : A \quad C[y] : \Gamma, B^y \vdash \ldots, S'_i : \Delta_i^{\delta_i}, \ldots (, S'_{n+1} : C^{\beta}); S' : C}{\textbf{make-coroutine}\ t\ C_{\beta} : \Gamma \vdash \ldots, S_i \cup S'_i : \Delta_i^{\delta_i}, \ldots, S_{n+1} \cup S'_{n+1} \cup S'[S/y] : C^{\beta}; S : A - B}(I_-)$$

Again, by induction hypothesis, $S_i = S_{\delta_i}(t)$, $S'_i = S_{\delta_i}(C[y]) = S_{\delta_i}(C[\ ])$, $S_{n+1} = \mathcal{S}_{\beta}(t)$, $S'_{n+1} = \mathcal{S}_{\beta}(C[y]) = \mathcal{S}_{\beta}(C[\,])$, $S'[S/y] = \mathcal{S}_{[]}(C[y])[S/y] = \mathcal{S}_{[]}(C[\,]) \cup S$ and by definition 5.1:

$$\begin{array}{rcl} \mathcal{S}_{[]}(\textbf{make-coroutine}\ t\ C_{\beta}) & = & \mathcal{S}_{[]}(t) \ = \ S \\ \mathcal{S}_{\beta}(\textbf{make-coroutine}\ t\ C_{\beta}) & = & \mathcal{S}_{[]}(t) \cup \mathcal{S}_{[]}(C[\,]) \cup \mathcal{S}_{\beta}(t) \cup \mathcal{S}_{\beta}(C[\,]) \ = S'[S/y] \cup S_{n+1} \cup S'_{n+1} \\ \mathcal{S}_{\delta_i}(\textbf{make-coroutine}\ t\ C_{\beta}) & = & \mathcal{S}_{\delta_i}(t) \cup \mathcal{S}_{\delta_i}(C[\,]) \ = \ S_i \cup S'_i. \end{array}$$

∎

COROLLARY 5.10
Given a derivation of the typing judgement $t : \Gamma \vdash \Delta$ in CND$_{\rightarrow\vee\wedge-}$, if t is safe with respect to first-class coroutines then the derivation of $t : \Gamma \vdash \Delta$ belongs to SND$_{\rightarrow\vee\wedge-}$ (and is thus valid in subtractive logic).

PROOF. Let us consider an occurrence of the introduction rule for implication:

$$\frac{u : \Gamma, A^x \vdash \Delta_1^{\alpha_1}, \ldots, \Delta_m^{\alpha_m}; B}{\lambda x.u : \Gamma \vdash \Delta_1^{\alpha_1}, \ldots, \Delta_m^{\alpha_m}; A \rightarrow B}$$

Since t is safe with respect to coroutine contexts, for any α_j, $x \notin \mathcal{S}_{\alpha_j}(u)$ and then by Theorem 5.9, there is no link between A^x and Δ, thus this occurrence of the introduction rule for implication is constructive. Let us now consider an occurrence of the elimination rule for subtraction:

$$\frac{t : \Gamma \vdash \Delta; A - B \quad u : \Gamma, A^x \vdash \Delta; B}{\textbf{resume}\ t\ \textbf{with}\ x \mapsto u : \Gamma \vdash \Delta; C}$$

Since *t* is safe with respect to first-class coroutines, we know that $\mathcal{S}_{[]}(u) \subseteq \{x\}$ and then by Theorem 5.9, there is no link between Γ and B, thus this occurrence of the elimination rule for subtraction is constructive. ∎

COROLLARY 5.11
Given a derivation of the typing judgement $t : \Gamma \vdash \Delta$ in CND$_{\rightarrow\vee\wedge}$, if t is safe with respect to coroutine contexts then the derivation of $t : \Gamma \vdash \Delta$ belongs to SND$_{\rightarrow\vee\wedge}$ (and is thus valid in intuitionistic logic).

EXAMPLE 5.12
Let us decorate the proof of example 3.20 by $\lambda\mu^{\rightarrow+\times-}$-terms, we obtain:

$$\dfrac{\dfrac{\dfrac{z : A \vee B^z \vdash; A \vee B \quad \overline{\textbf{set-context}\ \alpha\ x : A^x \vdash A^\alpha; B} \quad y : B^y \vdash; B}{\textbf{case}\ z\ \textbf{of}\ (\textbf{inl}\ x) \mapsto \textbf{set-context}\ \alpha\ x \mid (\textbf{inr}\ y) \mapsto y : A \vee B^z \vdash A^\alpha; B} \quad w : C^w \vdash; C}{\langle \textbf{case}\ z\ \textbf{of}\ (\textbf{inl}\ x) \mapsto \textbf{set-context}\ \alpha\ x \mid (\textbf{inr}\ y) \mapsto y, w\rangle : A \vee B^z, C^w \vdash A^\alpha; B \wedge C}}{\lambda w.\, \langle \textbf{case}\ z\ \textbf{of}\ (\textbf{inl}\ x) \mapsto \textbf{set-context}\ \alpha\ x \mid (\textbf{inr}\ y) \mapsto y, w\rangle : A \vee B^z \vdash A^\alpha; C \rightarrow (B \wedge C)}$$

One can check that this term is safe with respect to coroutine contexts, since we have:

$$w \notin \mathcal{S}_\alpha(\langle \textbf{case}\ z\ \textbf{of}\ (\textbf{inl}\ x) \mapsto \textbf{set-context}\ \alpha\ x \mid (\textbf{inr}\ y) \mapsto y, w\rangle) = \{z\}$$

and consequently the derivation is valid in intuitionistic logic.

REMARK 5.13
Note that the proof-term which decorates the conclusion in the previous example (let us call it t) is not a weakening term according to the Definition 11 given by Ritter, Pym and Wallen in [41] since otherwise $A \vee B \vdash A$ should be derivable (by Lemma 14), and the occurrence of α in t is not a weakening occurrence (according to the same definition) since otherwise $A \vee B \vdash C \rightarrow (B \wedge C)$ should be derivable. The term t is thus not intuitionistic according to Definition 13.

On the other hand we conjecture that any intuitionistic term (according to Definition 13 in [41]) is safe with respect to coroutine contexts. Consequently, our notion of 'safeness' is likely to allow more $\lambda\mu^{\rightarrow+\times}$-term as proof terms for sequents valid in intuitionistic logic.

6 Closure under reduction

The remainder of the paper is devoted to proving that the subset of safe $\lambda\mu^{\rightarrow+\times-}$-terms is closed under the reduction rules of the $\lambda\mu^{\rightarrow+\times-}$-calculus. We shall begin with Proposition 6.8 which says that the reduction rules of the $\lambda\mu^{\rightarrow+\times-}$-calculus do not provide new dependencies (in safe terms). We first need the following additional lemmas about substitutions and contexts:

LEMMA 6.1
$\mathcal{S}_\delta(t) = \emptyset$ if δ does not occur in t.

LEMMA 6.2
If u and v are $\lambda\mu^{\rightarrow+\times-}$-terms then:

$$\begin{array}{lcl} \mathcal{S}_{[]}(u\{v/x\}) & \subseteq & \mathcal{S}_{[]}(u)[\mathcal{S}_{[]}(v)/x] \\ \mathcal{S}_\delta(u\{v/x\}) & \subseteq & \mathcal{S}_\delta(u)[\mathcal{S}_{[]}(v)/x] \cup \mathcal{S}_\delta(v). \end{array}$$

LEMMA 6.3
If t is a $\lambda\mu^{\rightarrow+\times-}$-term then:

$$\begin{array}{lcl} \mathcal{S}_{[]}(t\{\beta/\alpha\}) & \subseteq & \mathcal{S}_{[]}(t) \\ \mathcal{S}_\beta(t\{\beta/\alpha\}) & \subseteq & \mathcal{S}_\beta(t) \cup \mathcal{S}_\alpha(t) \\ \mathcal{S}_\delta(t\{\beta/\alpha\}) & \subseteq & \mathcal{S}_\delta(t) \text{ for any } \delta \neq \beta. \end{array}$$

LEMMA 6.4
If t is a $\lambda\mu^{\rightarrow+\times-}$-terms and $C[\ ]$ is a simple $\lambda\mu^{\rightarrow+\times-}$-context such that α does not occur in $C[\ ]$ then:

$$\begin{array}{lcl}\mathcal{S}_{[]}(t\{\alpha \hookleftarrow C[\,]\}) & \subseteq & \mathcal{S}_{[]}(t)\\ \mathcal{S}_{\alpha}(t\{\alpha \hookleftarrow C[\,]\}) & \subseteq & \mathcal{S}_{\alpha}(t) \cup \mathcal{S}_{[]}(C[\,])\\ \mathcal{S}_{\delta}(t\{\alpha \hookleftarrow C[\,]\}) & \subseteq & \mathcal{S}_{\delta}(t) \cup \mathcal{S}_{\delta}(C[\,]) \text{ for any } \delta \neq \alpha.\end{array}$$

LEMMA 6.5
Let u be a $\lambda\mu^{\rightarrow+\times-}$-term, let $C[\,]$ be an simple $\lambda\mu^{\rightarrow+\times-}$-context, and let δ be a free name in $C[u]$:

$$\begin{array}{l}\mathcal{S}_{[]}(C[u]) = \mathcal{S}_{[]}(C[\,]) \cup \mathcal{S}_{[]}(u)\\ \mathcal{S}_{\delta}(C[u]) = \mathcal{S}_{\delta}(C[\,]) \cup \mathcal{S}_{\delta}(u).\end{array}$$

LEMMA 6.6
Given an instance $r \rightsquigarrow s$ of a rule of the $\lambda\mu^{\rightarrow+\times-}$-calculus, if r is safe then $\mathcal{S}_{[]}(s) \subseteq \mathcal{S}_{[]}(r)$ and $\mathcal{S}_{\delta}(s) \subseteq \mathcal{S}_{\delta}(r)$ for any a free name δ of s.

PROOF. Let us consider each reduction rule of the $\lambda\mu^{\rightarrow+\times-}$-calculus. Let y be a free variable of s (and thus a free variable of r) and let δ be a free name in s (and thus a free name in r):

Detour-reduction

(a) $r = (\lambda x.u\ v)$ and $s = u\{v/x\}$. By Lemma 6.2:
- $\mathcal{S}_{[]}(u\{v/x\}) \subseteq \mathcal{S}_{[]}(u)[\mathcal{S}_{[]}(v)/x] \subseteq \mathcal{S}_{[]}(u)\backslash\{x\} \cup \mathcal{S}_{[]}(v) = \mathcal{S}_{[]}(\lambda x.u\ v)$
- $\mathcal{S}_{\delta}(u\{v/x\}) \subseteq \mathcal{S}_{\delta}(u)[\mathcal{S}_{[]}(v)/x] \cup \mathcal{S}_{\delta}(v) = \mathcal{S}_{\delta}(u) \cup \mathcal{S}_{\delta}(v) = \mathcal{S}_{\delta}(\lambda x.u\ v)$
  (since r is safe and thus $x \notin \mathcal{S}_{\delta}(u)$).

(b) $r = \mathbf{fst}\ \langle t, u\rangle$ and $s = t$.
- $\mathcal{S}_{[]}(t) \subseteq \mathcal{S}_{[]}(t) \cup \mathcal{S}_{[]}(u) = \mathcal{S}_{[]}(\mathbf{fst}\ \langle t, u\rangle)$
- $\mathcal{S}_{\delta}(t) \subseteq \mathcal{S}_{\delta}(t) \cup \mathcal{S}_{\delta}(u) = \mathcal{S}_{\delta}(\mathbf{fst}\ \langle t, u\rangle)$.

(c) Similar to (b).

(d) $r = \mathbf{case}\ (\mathbf{inl}\ t)\ \mathbf{of}\ (\mathbf{inl}\ x) \mapsto u \mid (\mathbf{inr}\ y) \mapsto v$ and $s = u\{t/x\}$. By Lemma 6.2:
- $\mathcal{S}_{[]}(u\{t/x\}) \subseteq \mathcal{S}_{[]}(u)[\mathcal{S}_{[]}(t)/x] \subseteq \mathcal{S}_{[]}(u)[\mathcal{S}_{[]}(t)/x] \cup \mathcal{S}_{[]}(v)[\mathcal{S}_{[]}(t)/y] = \mathcal{S}_{[]}(r)$
- $\mathcal{S}_{\delta}(u\{t/x\}) \subseteq \mathcal{S}_{\delta}(u)[\mathcal{S}_{[]}(t)/x] \cup \mathcal{S}_{\delta}(t)$
  $\subseteq \mathcal{S}_{\delta}(u)[\mathcal{S}_{[]}(t)/x] \cup \mathcal{S}_{\delta}(v)[\mathcal{S}_{[]}(t)/y] \cup \mathcal{S}_{\delta}(t) = \mathcal{S}_{\delta}(r)$.

(e) Similar to (d).

(f) $r = u\{v/x\}$ and $s = \mathbf{let}\ x = v\ \mathbf{in}\ u$. By Lemma 6.2:
- $\mathcal{S}_{[]}(u\{v/x\}) \subseteq \mathcal{S}_{[]}(u)[\mathcal{S}_{[]}(v)/x] = \mathcal{S}_{[]}(\mathbf{let}\ x = v\ \mathbf{in}\ u)$
- $\mathcal{S}_{\delta}(u\{v/x\}) \subseteq \mathcal{S}_{\delta}(u)[\mathcal{S}_{[]}(v)/x] \cup \mathcal{S}_{\delta}(v) = \mathcal{S}_{\delta}(\mathbf{let}\ x = v\ \mathbf{in}\ u)$.

(g) $r = \mathbf{resume}\ (\mathbf{make\text{-}coroutine}\ v\ C_{\alpha})\ \mathbf{with}\ x \mapsto u$ and
$s = \mathbf{set\text{-}context}\ \beta\ C[u\{v/x\}]$.
- $\mathcal{S}_{[]}(\mathbf{set\text{-}context}\ \beta\ C[u\{v/x\}]) = \emptyset$
  $\subseteq \mathcal{S}_{[]}(\mathbf{resume}\ (\mathbf{make\text{-}coroutine}\ v\ C_{\beta})\ \mathbf{with}\ x \mapsto u)$.
- $\mathcal{S}_{\beta}(\mathbf{set\text{-}context}\ \beta\ C[u\{v/x\}]) = \mathcal{S}_{\beta}(C[u\{v/x\}]) \cup \mathcal{S}_{[]}(C[u\{v/x\}])$
  $= \mathcal{S}_{\beta}(C[\,]) \cup \mathcal{S}_{\beta}(u\{v/x\}) \cup \mathcal{S}_{[]}(C[\,]) \cup \mathcal{S}_{[]}(u\{v/x\})$
  $\subseteq \mathcal{S}_{\beta}(C[\,]) \cup \mathcal{S}_{\beta}(u)[\mathcal{S}_{[]}(v)/x] \cup \mathcal{S}_{\beta}(v) \cup \mathcal{S}_{[]}(C[\,]) \cup \mathcal{S}_{[]}(u)[\mathcal{S}_{[]}(v)/x]$
  $\subseteq \mathcal{S}_{\beta}(C[\,]) \cup \mathcal{S}_{\beta}(u)[\mathcal{S}_{[]}(v)/x] \cup \mathcal{S}_{\beta}(v) \cup \mathcal{S}_{[]}(C[\,]) \cup \mathcal{S}_{[]}(v)$ since $\mathcal{S}_{[]}(u) \subseteq \{x\}$
  $= \mathcal{S}_{\beta}(u)[\mathcal{S}_{[]}(\mathbf{v})/x] \cup \big(\mathcal{S}_{\beta}(v) \cup \mathcal{S}_{\beta}(C[\,]) \cup \mathcal{S}_{[]}(v) \cup \mathcal{S}_{[]}(C[\,])\big)$
  $= \mathcal{S}_{\beta}(u)[\mathcal{S}_{[]}(\mathbf{v})/x] \cup \mathcal{S}_{\beta}(\mathbf{make\text{-}coroutine}\ v\ C_{\beta})$
  $= \mathcal{S}_{\beta}(u)[\mathcal{S}_{[]}(\mathbf{make\text{-}coroutine}\ v\ C_{\beta})/x] \cup \mathcal{S}_{\beta}(\mathbf{make\text{-}coroutine}\ v\ C_{\beta})$
  $= \mathcal{S}_{\beta}(\mathbf{resume}\ (\mathbf{make\text{-}coroutine}\ v\ C_{\beta})\ \mathbf{with}\ x \mapsto u)$.

- $\mathcal{S}_\delta(\textbf{set-context}\beta\ C[u\{v/x\}]) = \mathcal{S}_\delta(\mathcal{C}[u\{v/x\}])$
  $= \mathcal{S}_\delta(C[\,]) \cup \mathcal{S}_\delta(u\{v/x\})$
  $\subseteq \mathcal{S}_\delta(C[\,]) \cup \mathcal{S}_\delta(u)[\mathcal{S}_{[]}(v)/x] \cup \mathcal{S}_\delta(v)$
  $= \mathcal{S}_\delta(u)[\mathcal{S}_{[]}(v)/x] \cup (\mathcal{S}_\delta(v) \cup \mathcal{S}_\delta(C[\,]))$
  $= \mathcal{S}_\delta(u)[\mathcal{S}_{[]}(\mathbf{v})/x] \cup \mathcal{S}_\delta(\textbf{make-coroutine}\ v\ C_\beta)$
  $= \mathcal{S}_\delta(u)[\mathcal{S}_{[]}(\textbf{make-coroutine}\ v\ C_\beta)/x] \cup \mathcal{S}_\delta(\textbf{make-coroutine}\ v\ C_\beta)$
  $= \mathcal{S}_\delta(\textbf{resume}\ (\textbf{make-coroutine}\ v\ C_\beta)\ \textbf{with}\ x \mapsto u)$.

Structural reduction

(a) $r = C[\mu\alpha.u]$ and $s = \mu\alpha.u\{\alpha \hookleftarrow C[\,]\}$. By lemma 6.4 and lemma 6.5:
- $\mathcal{S}_{[]}(\mu\alpha.u\{\alpha \hookleftarrow C[\,]\}) = \mathcal{S}_\alpha(u\{\alpha \hookleftarrow C[\,]\}) = \mathcal{S}_\alpha(u) \cup \mathcal{S}_{[]}(C[\,])$
  $= \mathcal{S}_{[]}(C[\,]) \cup \mathcal{S}_{[]}(\mu\alpha.u) = \mathcal{S}_{[]}(C[\mu\alpha.u])$
- $\mathcal{S}_\delta(\mu\alpha.u\{\alpha \hookleftarrow C[\,]\}) \subseteq \mathcal{S}_\delta(u\{\alpha \hookleftarrow C[\,]\}) \subseteq \mathcal{S}_\delta(u) \cup \mathcal{S}_\delta(C[\,])$
  $= \mathcal{S}_\delta(C[\mu\alpha.u])$.

(b) $r = C[\textbf{case}\ t\ \textbf{of}\ (\textbf{inl}\ x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v]$ and
$s = \textbf{case}\ t\ \textbf{of}\ (\textbf{inl}\ x) \mapsto C[u] \mid (\textbf{inr}\ y) \mapsto C[v]$
- $\mathcal{S}_{[]}(\textbf{case}\ t\ \textbf{of}\ (\textbf{inl}\ x) \mapsto C[u] \mid (\textbf{inr}\ y) \mapsto C[v])$
  $= \mathcal{S}_{[]}(C[u])[\mathcal{S}_{[]}(t)/x] \cup \mathcal{S}_{[]}(C[v])[\mathcal{S}_{[]}(t)/y]$
  $= \mathcal{S}_{[]}(C[\,]) \cup \mathcal{S}_{[]}(u)[\mathcal{S}_{[]}(t)/x] \cup \mathcal{S}_{[]}(v)[\mathcal{S}_{[]}(t)/y]$
  $= \mathcal{S}_{[]}(C[\textbf{case}\ t\ \textbf{of}\ (\textbf{inl}\ x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v])$
- $\mathcal{S}_\delta(\textbf{case}\ t\ \textbf{of}\ x \mapsto C[u] \mid y \mapsto C[v]) = \mathcal{S}_\delta(C[u])[\mathcal{S}_\delta(t)/x] \cup \mathcal{S}_\delta(C[v])[\mathcal{S}_\delta(t)/y]$
  $= \mathcal{S}_\delta(C[\,]) \cup \mathcal{S}_\delta(u)[\mathcal{S}_\delta(t)/x] \cup \mathcal{S}_\delta(v)[\mathcal{S}_\delta(t)/y]$
  $= \mathcal{S}_\delta(C[\textbf{case}\ t\ \textbf{of}\ (\textbf{inl}\ x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v])$.

(c) $r = C[\textbf{resume}\ t\ \textbf{with}\ x \mapsto u]$ and $s = \textbf{resume}\ t\ \textbf{with}\ x \mapsto u$
- $\mathcal{S}_{[]}(\textbf{resume}\ t\ \textbf{with}\ x \mapsto u) = \mathcal{S}_{[]}(C[u])[\mathcal{S}_{[]}(t)/x]$
  $= \mathcal{S}_{[]}(C[\,]) \cup \mathcal{S}_{[]}(u)[\mathcal{S}_{[]}(t)/x]$
  $\subseteq \mathcal{S}_{[]}(u)[\mathcal{S}_{[]}(t)/x]$
  $= \mathcal{S}_{[]}(C[\textbf{resume}\ t\ \textbf{with}\ x \mapsto u])$
- $\mathcal{S}_\delta(\textbf{resume}\ t\ \textbf{with}\ x \mapsto C[u]) = \mathcal{S}_\delta(C[u])[\mathcal{S}_{[]}(t)/x] \cup \mathcal{S}_\delta(t)$
  $= \mathcal{S}_\delta(C[\,]) \cup \mathcal{S}_\delta(u)[\mathcal{S}_{[]}(t)/x] \cup \mathcal{S}_\delta(t)$
  $\subseteq \mathcal{S}_\delta(u)[\mathcal{S}_{[]}(t)/x] \cup \mathcal{S}_\delta(t)$
  $= \mathcal{S}_\delta(C[\textbf{resume}\ t\ \textbf{with}\ x \mapsto u])$.

Simplification

(a) $r = \mu\alpha[\alpha]u$ and $s = u$ where α does not occur free in u.
- $\mathcal{S}_{[]}(u) = \mathcal{S}_{[]}(u) \cup \mathcal{S}_\alpha(u) = \mathcal{S}_\alpha([\alpha]u) = \mathcal{S}_{[]}(\mu\alpha[\alpha]u)$ (since $\mathcal{S}_\alpha(u) = \emptyset$)
- $\mathcal{S}_\delta(u) = \mathcal{S}_\delta([\alpha]u) = \mathcal{S}_\delta(\mu\alpha[\alpha]u)$.

(b) $r = [\beta]\mu\alpha.t$ and $s = t\{\beta/\alpha\}$. By Lemma 6.3:
- $\mathcal{S}_{[]}(t\{\beta/\alpha\}) \subseteq \mathcal{S}_{[]}(t) = \emptyset = \mathcal{S}_\alpha([\beta]\mu\alpha.t)$ (since t has the form $[\delta]v$)
- $\mathcal{S}_\beta(t\{\beta/\alpha\}) \subseteq \mathcal{S}_\beta(t) \cup \mathcal{S}_\alpha(t) = \mathcal{S}_\beta(\mu\alpha.t) \cup \mathcal{S}_{[]}(\mu\alpha.t) = \mathcal{S}_\beta([\beta]\mu\alpha.t)$
- $\mathcal{S}_\delta(t\{\beta/\alpha\}) \subseteq \mathcal{S}_\delta(t) = \mathcal{S}_\delta(\mu\alpha.t) = \mathcal{S}_\delta([\beta]\mu\alpha.t)$ for any $\delta \neq \beta$.

∎

LEMMA 6.7
Given two $\lambda\mu^{\rightarrow+\times-}$-terms t, u and an arbitrary context $C[\,]$, if $\mathcal{S}_{[]}(u) \subseteq \mathcal{S}_{[]}(t)$ and $\mathcal{S}_\delta(u) \subseteq \mathcal{S}_\delta(t)$ then $\mathcal{S}_{[]}(C[u]) \subseteq \mathcal{S}_{[]}(C[t])$ and $\mathcal{S}_\delta(C[u]) \subseteq \mathcal{S}_\delta(C[t])$ for any a free name δ of $C[t]$.

PROOF. By induction on the context $C[\,]$. ∎

PROPOSITION 6.8
Given a $\lambda\mu^{\rightarrow+\times-}$-term t, if t is safe and $t \rightsquigarrow u$ then $\mathcal{S}_{[]}(u) \subseteq \mathcal{S}_{[]}(t)$ and $\mathcal{S}_\delta(u) \subseteq \mathcal{S}_\delta(t)$ for any a free name δ of t.

PROOF. By Lemma 6.6 and Lemma 6.7. ∎

LEMMA 6.9
Given two $\lambda\mu^{\rightarrow+\times-}$-terms u, v, if u and v are safe w.r.t. coroutine contexts then $u\{v/x\}$ is safe w.r.t. coroutine contexts.

PROOF. Let $\lambda y.t$ be a subterm of $u\{v/x\}$. Either $\lambda y.t$ is a subterm of v or y does not occur in v. Consequently, $y \notin \mathcal{S}_\delta(t)$ since v is safe in the former case and since u is safe in the latter case. ∎

LEMMA 6.10
Given an instance $r \rightsquigarrow s$ of a rule of the $\lambda\mu^{\rightarrow+\times-}$-calculus, if r is safe w.r.t. coroutine contexts then s is safe w.r.t. coroutine contexts.

PROOF. Again, we consider each rule of the $\lambda\mu^{\rightarrow+\times-}$-calculus:

Detour-reduction

(a) $r = (\lambda x.u\ v)$ and $s = u\{v/x\}$. Apply Lemma 6.9.

(b) $r =$ **fst** $\langle t, u\rangle$ and $s = t$. Then $s = t$ is safe.

(c) Similar to (b).

(d) $r =$ **case** (**inl** t) **of** (**inl** x) $\mapsto u \mid$ (**inr** y) $\mapsto v$ and $s = u\{t/x\}$. Apply Lemma 6.9.

(e) Similar to (d).

(f) $r = u\{v/x\}$ and $s =$ **let** $x = v$ **in** u. Apply Lemma 6.9.

(g) $r =$ **resume** (**make-coroutine** $v\ C_\alpha$) **with** $x \mapsto u$ and $s =$ **set-context** $\beta\ C[u\{v/x\}]$. By Lemma 6.9, $u\{v/x\}$ is safe w.r.t. coroutine contexts, and since $C[\ ]$ is safe w.r.t. coroutine contexts, we know by Lemma 6.7 that s is also safe w.r.t. coroutine contexts.

Structural reduction

(a) $r = C[\mu\alpha.w]$ and $s = \mu\alpha.w\{\alpha \leftarrow C[\,]\}$.
Let $\lambda y.t$ be a subterm of $\mu\alpha.w\{\alpha \leftarrow C[\,]\}$, either $\lambda y.t$ is a subterm of $C[\,]$ or y does not occur in $C[\,]$. Consequently, $y \notin \mathcal{S}_\delta(t)$ since $C[\,]$ is safe in the former case and since w is safe in the latter case.

(b) $r = C[$**case** w **of** (**inl** x) $\mapsto u \mid$ (**inr** y) $\mapsto v]$ and
$s =$ **case** w **of** (**inl** x) $\mapsto C[u] \mid$ (**inr** y) $\mapsto C[v]$.
Let $\lambda y.t$ be a subterm of **case** w **of** (**inl** x) $\mapsto C[u] \mid$ (**inr** y) $\mapsto C[v]$, either $\lambda y.t$ is a subterm of $C[\,]$ or y does not occur in $C[\,]$. Consequently, $y \notin \mathcal{S}_\delta(t)$ since $C[\,]$ is safe in the former case and since u, v and w are safe in the latter case.

(c) $r = C[$**resume** c **with** $x \mapsto u]$ and $s =$ **resume** c **with** $x \mapsto u$.
Obviously s is safe w.r.t. coroutine contexts since it is a subterm of r.

Simplification

(a) $r = \mu\alpha[\alpha]u$ and $s = u$ where α does not occur free in u. Then $s = u$ is safe.

(b) $r = [\beta]\mu\alpha.u$ and $s = u\{\beta/\alpha\}$. Let $\lambda y.t$ be a subterm of $u\{\beta/\alpha\}$ and let $\lambda y.v$ be the subterm of u such as $\lambda y.v = \lambda y.t\{\beta/\alpha\}$. Then $y \notin \mathcal{S}_\delta(t) = \mathcal{S}_\delta(v)$ and $y \notin \mathcal{S}_\beta(t) = \mathcal{S}_\beta(v) \cup \mathcal{S}_\alpha(v)$ since u is safe.

∎

LEMMA 6.11
Given two $\lambda\mu^{\rightarrow+\times-}$-terms t, u safe w.r.t. first-class coroutines and such that $\mathcal{S}_{[]}(u) \subseteq \mathcal{S}_{[]}(t)$ and $\mathcal{S}_\delta(u) \subseteq \mathcal{S}_\delta(t)$ and an arbitrary context $C[\ ]$, if $C[t]$ is safe then $C[u]$ is safe w.r.t. first-class coroutines.

PROOF. By induction on the context $C[\ ]$. ∎

LEMMA 6.12
Given an instance $r \rightsquigarrow s$ of a rule of the $\lambda\mu^{\rightarrow+\times-}$-calculus, if r is safe w.r.t. first-class coroutines then s is safe w.r.t. first-class coroutines.

PROOF. We already know by Lemma 6.10 that s is safe w.r.t. coroutine contexts. We just have to check that for any subterm of s which has the form **resume** c **with** $x \mapsto u$, $\mathcal{S}_{[]}(u) \subseteq \{x\}$. This property follows from Lemma 6.6 and Lemma 6.7. ∎

THEOREM 6.13
Given a $\lambda\mu^{\rightarrow+\times-}$-term t, if t is safe with respect to first-class coroutines and $t \rightsquigarrow u$ then u is safe with respect to first-class coroutines.

PROOF. Let us write t as $C[r]$, where r is the redex to be reduced and let $r \rightsquigarrow s$ be the instance of the rule which is applied. By Lemma 6.12, s is safe w.r.t. first-class coroutines, and by Lemma 6.6, we have $\mathcal{S}_{[]}(s) \subseteq \mathcal{S}_{[]}(r)$ and $\mathcal{S}_\delta(s) \subseteq \mathcal{S}_\delta(r)$ for any a free name δ of s. Then, by Lemma 6.11 u is safe w.r.t. first-class coroutines. ∎

COROLLARY 6.14
Given a $\lambda\mu^{\rightarrow+\times}$-term t, if t is safe with respect to coroutine contexts and $t \rightsquigarrow u$ then u is safe with respect to coroutine contexts.

7 Conclusion and further work

We have defined the safe $\lambda\mu^{\rightarrow+\times}$-calculus, which is closed under reduction and whose type system corresponds to intuitionistic logic. In this calculus, continuations are not first-class objects but the ability of context-switching remains. The safe $\lambda\mu^{\rightarrow+\times-}$-calculus is an extension of the $\lambda\mu^{\rightarrow+\times}$-calculus with first-class coroutines. First-class coroutines are strictly less powerful than first-class continuations: the type system of the $\lambda\mu^{\rightarrow+\times-}$-calculus corresponds to subtractive logic, which is conservative over intuitionistic logic.

We have proved that first-class coroutines disappear during the normalization process (since the subformula property holds for normal forms) whenever the type of the term contains no subtraction (see Proposition 4.9). The normal form belongs to the safe $\lambda\mu^{\rightarrow+\times}$-calculus. If we extend our work to the first-order framework, and since subtractive logic is conservative over CDL (and the existence property holds in CDL), we expect to be able to extract witnesses from normal proofs of existential formulas (which contain no subtraction). A first attempt could consist in exploiting the proof of conservativity given in Appendix B. Another (better) solution would be to derive the

existence property from the subformula property (however this is not straightforward in a deduction system with multi-conclusioned sequents). A forthcoming paper shall be devoted to this issue.

Applications of first-class coroutines were barely mentioned in this paper. In fact, practical applications of coroutines often use other extensions such as imperative features which do not easily fit in the formulae-as-types framework. However, purely functional examples have to be explored. On the other hand, it would be interesting to define an abstract machine for the safe $\lambda\mu^{\rightarrow+\times}$-calculus and to investigate what kind of optimization is allowed by the 'safeness' property (as opposed to full-fledged continuations).

Eventually, the duality call-by-name/call-by-value (from the classical $\lambda\mu$-calculus) should be revisited in our framework, where the duality is likely to exchange functions and coroutines. We already know by construction of the safe $\lambda\mu^{\rightarrow+\times-}$-calculus that the dual of a safe $\lambda\mu^{\rightarrow+\times-}$-term is also safe.

Acknowledgements

I am very grateful to Serge Grigorieff for his careful reading and to Hugo Herbelin for helpful comments to earlier versions of this article. I also wish to thank one anonymous referee for his valuable report and his suggestions which helped to improve the manuscript in many ways.

References

[1] Z. Ariola and H. Herbelin. Minimal classical logic and control operators. In *Proceedings of ICALP'03*, volume 2719 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[2] F. Barbanera and S. Berardi. Extracting constructive content from classical logic via control-like reductions. In volume 662 of *Lecture Notes in Computer Science*, pp. 47–59. Springer-Verlag, 1994.

[3] E. W. Beth. Semantic construction of intuitionistic logic. *Koninklijke Nederlandse Akademie van Wetenschappen, Mededelingen, Nieuwe Reeks*, **19**, 357–388, 1956.

[4] T. Brauner and V. de Paiva. A formulation of linear logic based on dependency-relations. In *Proceedings of Annual Conference of the European Association for Computer Science Logic*, volume 1414 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.

[5] E. C. Cooper and J. G. Morrisett. Adding threads to standard ML. Report CMU-CS-90-186, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1990.

[6] T. Crolard. Extension de l'isomorphisme de Curry-Howard au traitement des exceptions (application d'une étude de la dualité en logique intuitionniste). Thèse de Doctorat. Université Paris 7, 1996.

[7] T. Crolard. A confluent lambda-calculus with a catch/throw mechanism. *Journal of Functional Programming*, **9**, 625–647, 1999.

[8] T. Crolard. Subtractive Logic. *Theoretical Computer Science*, **254**, 151–185, 2001.

[9] P.-L. Curien and H. Herbelin. The duality of computation. In *Proceedings of the ACM Sigplan International Conference on Functional Programming (ICFP-00)*, volume 35.9 of *ACM Sigplan Notices*, pp. 233–243, N.Y., September 18–21 2000. ACM Press.

[10] P. De Groote. On the relation between the lambda-calculus and the syntactic theory of sequential control. Volume 822 of *Lecture Notes in Computer Science*, pp. 31–43. Springer-Verlag, 1994.

[11] P. de Groote. A simple calculus of exception handling. In *Second International Conference on Typed Lambda Calculi and Applications*, Edinburgh. Volume 902 of *Lecture Notes in Computer Science*, pp. 201–215. Springer-Verlag, 1995.

[12] P. de Groote. Strong normalization of classical natural deduction with disjunction. In *Typed Lambda Calculi and Applications*, volume 2044 of *Lecture Notes in Computer Science*, p. 182. Springer-Verlag, 2001.

[13] V. de Paiva and E. Ritter. A Parigot-style linear lambda-calculus for full intuitionistic linear logic. Submitted, 2003.

[14] A. G. Dragalin. Mathematical intuitionism: introduction to proof theory. In *Translations of Mathematical Monographs*, volume 67. American Matematical Society, Providence, Rhode Island, 1988.

[15] B. Duba, R. Harper, and D. MacQueen. Typing first-class continuations in ML. In ACM-SIGPLAN ACM-SIGACT, *Conference Record of the 18th Annual ACM Symposium on Principles of Programming Languages (POPL '91)*, pp. 163–173, Orlando, FL, USA, January 1991. ACM Press.

[16] R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *The Journal of Symbolic Logic*, **57**, 795–807, 1992.

[17] A. Filinski. Declarative Continuations: An Investigation of Duality in Programming Language Semantics. In *Category Theory and Computer Science*, volume 389 of *Lecture Notes in Computer Science*, pp. 224–249. Springer-Verlag, 1989.

[18] D. P. Friedman, C. T. Haynes, and M. Wand. Continuations and coroutines: An exercise in metaprogramming. In *Proceedings of 1984 ACM Symposium on Lisp and Functional Programming*, pp. 293–298, August 1984.

[19] D. P. Friedman, C. T. Haynes, and M. Wand. Obtaining coroutines with continuations. *Journal of Computer Languages*, **11**, 143–153, 1986.

[20] D. M. Gabbay. *Semantical Investigations in Heyting's Intuitionistic Logic*. Reidel, Dordrecht, 1981.

[21] R. Goré. Dual intuitionistic logic revisited. In *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX 2000, St Andrews, Scotland, UK, July 3-7, 2000, Proceedings*, Roy Dyckhoff, ed., volume 1847 of *Lecture Notes in Computer Science*, pp. 252–267. Springer, 2000.

[22] S. Görnemann. A logic stronger than intuitionism. *The Journal of Symbolic Logic*, **36**, 249–261, 1971.

[23] T. G. Griffin. A formulæ-as-type notion of control. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Langages*, pp. 47–58, 1990.

[24] The Open Group. The Single UNIX Specification, Version 2, 1997. Available from http://www.UNIX-systems.org/online.html.

[25] M. Hyland and V. de Paiva. Full intuitionistic linear logic (extended abstract). *Annals of Pure and Applied Logic*, **64**, 273–291, 1993.

[26] Y. Kameyama. A new formulation of the catch/throw mechanism. In *Second Fuji International Workshop on Functional and Logic Programming*, T. Ida, A. Ohori, and M. Takeichi, eds, pp. 106–122. Word Scientific, 1997.

[27] Y. Kameyama and M. Sato. A classical catch/throw calculus with tag abstraction and its strong normalizability. In *Proceedings of the fourth Australasian Theory Symposium*, volume 20-3 of *Australian Computer Science Communications*, X. Lin, ed., pp. 183–197. Springer-Verlag, 1998.

[28] Y. Kameyama and M. Sato. Strong normalizability of the non-deterministic catch/throw calculi. *Theoretical Computer Science*, **272**, 223–245, 2002.

[29] S. C. Kleene. *Introduction to metamathematics*. North-Holland, Amsterdam, 1952. (Eighth reprint 1980.)

[30] J.-L. Krivine. Classical logic, storage operators and second order λ-calculus. *Annals of Pure and Applied Logic*, **68**, 53–78, 1994.

[31] C. R. Murthy. Classical proofs as programs: How, when, and why. Technical Report 91-1215, Cornell University, Department of Computer Science, 1991.

[32] H. Nakano. A constructive logic behind the catch and throw mechanism. *Annals of Pure and Applied Logic*, **69**, 269–301, 1994.

[33] H. Nakano. The non-deterministic catch and throw mechanism and its subject reduction property. In *Logic, Language and Computation*, volume 592 of *Lecture Notes in Computer Science*, pp. 61–72. Springer-Verlag, 1994.

[34] H. Nakano. *The Logical Structures of the Catch and Throw Mechanism*. PhD thesis, The University of Tokyo, 1995.

[35] H. Ono. Model extension theorem and Graigs interpolation theorem for intermediate predicate logics. *Rep. Math. Logic*, 15:41–57, 1983.

[36] M. Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Proceedings Logic Programming and Automated Reasoning*, volume 624 of *Lecture Notes in Computer Science*, pp. 190–201. Springer-Verlag, 1992.

[37] M. Parigot. Classical proofs as programs. In *Computationnal logic and theory*, volume 713 of *Lecture Notes in Computer Science*, pp. 263–276. Springer-Verlag, 1993.

[38] M. Parigot. Strong normalization for second order classical natural deduction. In *Proceedings of the Eighth Annual IEEE Symposium on Logic in Computer Science*, 1993.

[39] D. Pym and E. Ritter. On the semantics of classical disjunction. *Journal of Pure and Applied Algebra*, **159**, 315–338, 2001.

[40] D. Pym, E. Ritter, and L. Wallen. Proof-terms for classical and intuitionistic resolution. In *Proceedings of the Thirteenth International Conference on Automated Deduction (CADE-96)*, M. A. McRobbie and J. K. Slaney, eds, volume 1104 of *Lecture Notes in Artificial Intelligence*, pp. 17–31, Springer-Verlag, 1996.

[41] D. Pym, E. Ritter, and L. Wallen. On the intuitionistic force of classical search. *Theoretical Computer Science*, **232**, 299–333, 2000.

[42] N. Ramsey. Concurrent programming in ML. Technical Report CS-TR-262-90, Department of Computer Science, Princeton University, Princeton, NJ, 1990.

[43] C. Rauszer. A formalization of the propositional calculus of H-B logic. *Studia Logica*, **33**, 23–34, 1974.

[44] C. Rauszer. Semi-boolean algebras and their applications to intuitionistic logic with dual operations. In *Fundamenta Mathematicae*, **83**, 219–249, 1974.

[45] C. Rauszer. An algebraic and Kripke-style approach to a certain extension of intuitionistic logic. In *Dissertationes Mathematicae*, volume 167. Institut Mathématique de l'Académie Polonaise des Sciences, 1980.

[46] N. J. Rehof and M. H. Sørensen. The λ_Δ-calculus. In *Theoretical Aspects of Computer Software*, volume 542 of *Lecture Notes in Computer Science*, pp. 516–542. Springer-Verlag, 1994.

[47] J. H. Reppy. Asynchronous signals is standard ML. Technical Report TR90-1144, Cornell University, Computer Science Department, August 1990.

[48] J. H. Reppy. First-class synchronous operations, volume 907 of *Lecture Notes in Computer Science*, pp. 235–252, 1995.

[49] G. Restall. Extending intuitionistic logic with subtraction. Available as http://consequently.org/papers/extendingj.pdf, 1997.

[50] P. Selinger. Control categories and duality: On the categorical semantics of the lambda-mu calculus. *Mathematical Structures in Computer Science*, **11**, 207–260, 2001.

[51] M. E. Szabo. *Gentzen Collected work*. North-Holland, Amsterdam, 1969.

[52] P. Wadler. Call-by-value is dual to call-by-name. In *Proceedings of the Eighth ACM SIGPLAN International Conference on Functional Programming*, Uppsala, Sweden, August 2003. ACM Press.

[53] M. Wand. Continuation-based multiprocessing. In *Conference Record of the 1980 LISP Conference*, J. Allen, ed., pp. 19–28. The Lisp Company, 1980. Republished by ACM.

Appendix

A Properties of the $\lambda\mu^{\rightarrow+\times-}$-calculus

We shall take advantage of the definability of subtraction in classical logic to derive strong normalization and uniqueness of normal forms for the $\lambda\mu^{\rightarrow+\times-}$-calculus from the same properties of the $\lambda\mu^{\rightarrow+\times}$-calculus. In order to derive also the permutative reduction rules for the $\lambda\mu^{\rightarrow+\times-}$-calculus, we shall need a $\lambda\mu^{\rightarrow+\times}$-calculus with pattern matching: we thus add a new 'tensor' product $\otimes$ to our type system. Then we shall be able to define $A - B$ as $A \otimes \neg B$ and then **make-coroutine/resume** as macro-definitions. Recall that these macros are in fact extracted from our derivations of the introduction/elimination rules of the (defined) subtraction.

In [12], de Groote presents a proof of strong normalization of CND with primitive disjunction (and primitive conjunction with projections). We show here how to adapt this proof to take into account the rule for pattern matching. Although de Groote did not consider the simplification rules in his paper, it is easy to show that simplification (alone) is strongly normalizing and also that simplification may be postponed with respect with the other reduction relations.

De Groote's proof of strong normalization is two-fold: first he proves the strong normalization of the structural reductions (for untyped $\lambda\mu^{\rightarrow\wedge\vee\perp}$-terms) then he proves the strong normalization of typed $\lambda\mu^{\rightarrow\wedge\vee\perp}$-terms using a CPS-simulation. The latter proof is given in a propositional framework, but de Groote claims that since this CPS-translation is defined on the untyped $\lambda\mu$-terms, it may be raised to the second-order.

A.1 The $\lambda\mu^{\rightarrow+\times\otimes\perp}$-calculus

We add two term constructors to the syntax of the raw $\lambda\mu^{\rightarrow+\times}$-calculus:

$$M \quad ::= \quad \ldots \mid \quad (M, N) \mid \quad \textbf{match } M \textbf{ with } (x, y) \mapsto N.$$

We also define the simple $\lambda\mu^{\rightarrow+\times\otimes\perp}$-contexts by extending the grammar of simple $\lambda\mu^{\rightarrow+\times}$-contexts:

$$C \quad ::= \quad \ldots \mid \quad \textbf{match } M \textbf{ with } (x, y) \mapsto N.$$

REMARK A.1
Recall that **abort** t is defined in the $\lambda\mu$-calculus as **set-context** $\varepsilon\ t$ where ε is a free name. Consequently, we shall not consider here **abort** as a primitive instruction.

A.1.1 Reduction rules

We extend the reduction rules of the $\lambda\mu^{\rightarrow+\times}$-calculus with this new detour-reduction rule:
(g) **match** (u, v) **with** $(x, y) \mapsto t \quad \rightsquigarrow \quad t\{u/x, y/v\}$
and the structural reduction is now defined by the following rules:
(a) $C[\mu\alpha.u] \quad \rightsquigarrow \quad \mu\alpha.u\{\alpha \hookleftarrow C[\,]\}$
(b) $C[$**case** t **of** $(\textbf{inl}\ x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v] \quad \rightsquigarrow \quad$ **case** t **of** $(\textbf{inl}\ x) \mapsto C[u] \mid (\textbf{inr}\ y) \mapsto C[v]$
(c) $C[$**match** t **with** $(x, y) \mapsto u] \quad \rightsquigarrow \quad$ **match** t **with** $(x, y) \mapsto C[u]$
where $C[\,]$ ranges over simple $\lambda\mu^{\rightarrow+\times\otimes\perp}$-contexts.

A.1.2 Typing rules for $\otimes$

$$\frac{t : \Gamma \vdash \Delta; A \quad u : \Gamma \vdash \Delta; B}{(t, u) : \Gamma \vdash \Delta; A \otimes B}(\otimes_I) \qquad \frac{t : \Gamma \vdash \Delta; A \otimes B \quad u : \Gamma, A^x, B^y \vdash \Delta; C}{\textbf{match}\ t\ \textbf{with}\ (x, y) \mapsto u : \Gamma \vdash \Delta; C}(\otimes_E)$$

A.1.3 Strong normalization of the structural reductions

We provide $\lambda\mu^{\rightarrow+\times\otimes\perp}$-terms with a norm adapted from the norm introduced in [12] (we recall the full definitions but the new cases are only the two last of each definition):

DEFINITION A.2
The norm $|\cdot|$ assigned to the $\lambda\mu^{\rightarrow+\times\otimes\perp}$-terms is inductively defined as follows:
(a) $|x| = 1$
(b) $|\lambda x.t| = |t|$
(c) $|(t\ u)| = |t| + \#t \times |u|$
(d) $|\langle t, u\rangle| = |t| + |u|$
(e) $|\textbf{fst}\ t| = |t| + \#t$
(f) $|\textbf{snd}\ t| = |t| + \#t$
(g) $|\textbf{inl}\ t| = |t|$
(h) $|\textbf{inr}\ t| = |t|$
(i) $|\textbf{case}\ t\ \textbf{of}\ (\textbf{inl}\ x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v| = |t| + \#t \times (|t| + |u|)$
(j) $|\mu\alpha.t| = |t|$
(k) $|[\alpha]t| = |t|$
(l) $|(t, u)| = |t| + |u|$
(m) $|\textbf{match}\ t\ \textbf{with}\ (x, y) \mapsto u| = |t| + 2 \times \#t \times |u|$
where:
(a) $\#x = 1$
(b) $\#\lambda x.t = 1$
(c) $\#(t\ u) = \#t$
(d) $\#\langle t,\ u\rangle = 1$
(e) $\#\,\textbf{fst}\ t = \#t$
(f) $\#\,\textbf{snd}\ t = \#t$
(g) $\#\,\textbf{inl}\ t = 1$
(h) $\#\,\textbf{inr}\ t = 1$
(i) $\#\textbf{case}\ t\ \textbf{of}\ (\textbf{inl}\ x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v = (2 \times \#t) \times (\#u + \#v)$
(j) $\#\mu\alpha.t = \lfloor t \rfloor_\alpha$
(k) $\#[\alpha]t = 1$
(l) $\#(t,\ u) = 1$
(m) $\#\textbf{match}\ t\ \textbf{with}\ (x, y) \mapsto u = (2 \times \#t) \times (2 \times \#u)$
and where:
(a) $\lfloor x \rfloor_\alpha = 0$
(b) $\lfloor \lambda x.t \rfloor_\alpha = \lfloor t \rfloor_\alpha$

(c) $\lfloor (t\ u) \rfloor_\alpha = \lfloor t \rfloor_\alpha + \#t \times \lfloor u \rfloor_\alpha$
(d) $\lfloor \langle t, u \rangle \rfloor_\alpha = \lfloor t \rfloor_\alpha + \lfloor u \rfloor_\alpha$
(e) $\lfloor \textbf{fst}\ t \rfloor_\alpha = \lfloor t \rfloor_\alpha$
(f) $\lfloor \textbf{snd}\ t \rfloor_\alpha = \lfloor t \rfloor_\alpha$
(g) $\lfloor \textbf{inl}\ t \rfloor_\alpha = \lfloor t \rfloor_\alpha$
(h) $\lfloor \textbf{inr}\ t \rfloor_\alpha = \lfloor t \rfloor_\alpha$
(i) $\lfloor \textbf{case}\ t\ \textbf{of}\ (\textbf{inl}\ x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v \rfloor_\alpha = \lfloor t \rfloor_\alpha + \#t \times (\lfloor u \rfloor_\alpha + \lfloor v \rfloor_\alpha)$
(j) $\lfloor \mu\alpha.t \rfloor_\alpha = \lfloor t \rfloor_\alpha$
(k) $\lfloor [\alpha]t \rfloor_\alpha = \lfloor t \rfloor_\alpha + \#t$
(l) $\lfloor [\beta]t \rfloor_\alpha = \lfloor t \rfloor_\alpha$
(m) $\lfloor (t, u) \rfloor_\alpha = \lfloor t \rfloor_\alpha + \lfloor u \rfloor_\alpha$
(n) $\lfloor \textbf{match}\ t\ \textbf{with}\ (x, y) \mapsto u \rfloor_\alpha = \lfloor t \rfloor_\alpha + 2 \times \#t \times \lfloor u \rfloor_\alpha$

LEMMA A.3
If t and u are two $\lambda\mu^{\rightarrow+\times\otimes\perp}$-terms and $t \rightsquigarrow_S u$ (where $\rightsquigarrow_S$ denotes the structural reduction) then $|t| > |u|$.

A.1.4 CPS-simulation

We adapt here de Groote's modified CPS-translation, which simulates the relation of detour-reduction by strict β-reduction and the relation of structural reduction by equality.

DEFINITION A.4
The modified CPS-translation $\overline{\overline{t}}$ of any $\lambda\mu^{\rightarrow+\times\otimes\perp}$-term is defined as:

$$\overline{\overline{t}} = \lambda k.(t\ :\ k)$$

where k is a fresh variable and where the infix operator ':' obeys the following definition:
(a) $x\ :k = (x\ k)$
(b) $\lambda x.t\ :k = (k\ \lambda x.\overline{\overline{t}})$
(c) $(t\ u)\ :k = t\ :\ \lambda m.m\ \overline{\overline{u}}\ k$
(d) $\langle t, u \rangle\ :k = (k\ \lambda m.(m\ \overline{\overline{t}}\ \overline{\overline{u}}))$
(e) $\textbf{fst}\ t\ :k = t\ :\lambda m.(m\ \lambda i.\lambda j.(i\ k))$
(f) $\textbf{snd}\ t\ :k = t\ :\lambda m.(m\ \lambda i.\lambda j.(j\ k))$
(g) $\textbf{inl}\ t\ :k = (k\ \lambda i.\lambda j.(i\ \overline{\overline{t}}))$
(h) $\textbf{inr}\ t\ :k = (k\ \lambda i.\lambda j.(j\ \overline{\overline{t}}))$
(i) $\textbf{case}\ t\ \textbf{of}\ (\textbf{inl}\ x) \mapsto u \mid (\textbf{inr}\ y) \mapsto v\ :k = t\ :\lambda m.(m\ \lambda x.(u\ k)\ \lambda y.(v\ k))$
(j) $\mu\alpha.t\ :k = (t\ :\ \lambda k.k)\{k/\alpha\}$ if α occurs free in t
(k) $\mu\alpha.t\ :k = \lambda\alpha.(t\ :\ \lambda k.k)\ k$ otherwise
(l) $[\alpha]t\ :k = t\ \alpha$
(m) $(t, u)\ :k = (k\ \lambda m.(m\ \overline{\overline{t}}\ \overline{\overline{u}}))$
(n) $\textbf{match}\ t\ \textbf{with}\ (x, y) \mapsto u\ :k = t\ :\lambda m.(m\ \lambda x\lambda y.(u\ k))$
where m, i, j are fresh variables.

LEMMA A.5
Let t and u be two $\lambda\mu^{\rightarrow+\times\otimes\perp}$-terms. If $t \rightsquigarrow_D u$ (where $\rightsquigarrow_D$ denotes the structural reduction) then $\overline{\overline{t}} \rightsquigarrow_\beta \overline{\overline{u}}$.

LEMMA A.6
Let t and u be two $\lambda\mu^{\rightarrow+\times\otimes\perp}$-terms. If $t \rightsquigarrow_S u$ then $\overline{\overline{t}} = \overline{\overline{u}}$.

THEOREM A.7
The well-typed $\lambda\mu^{\rightarrow+\times\otimes\perp}$-terms are strongly normalizable.

A.1.5 Church–Rosser property of the $\lambda\mu^{\rightarrow+\times\otimes\bot}$-calculus

The Church–Rosser property follows from the local confluence of the reductions, the strong normalization and Newman lemma.

A.2 Strong normalization and confluence of the $\lambda\mu^{\rightarrow+\times-}$-calculus

THEOREM A.8
The typed $\lambda\mu^{\rightarrow+\times-}$-calculus is strongly normalizing and enjoys the Church–Rosser property.

PROOF. Let us denote by Φ the translation from the $\lambda\mu^{\rightarrow+\times-}$-calculus into the $\lambda\mu^{\rightarrow+\times\otimes\bot}$-calculus defined by the following macro-definitions:

$$\begin{array}{rcl} \textbf{make-coroutine}\ \ t\ C_\alpha & \equiv & (t, \lambda z.\textbf{set-context}\ \alpha\ C[z]) \\ \textbf{resume}\ t\ \textbf{with}\ x \mapsto u & \equiv & \textbf{match}\ t\ \textbf{with}\ (x,k) \mapsto\ \textbf{abort}\ (k\ u). \end{array}$$

It is enough to check the following properties: (1) Φ is a morphism for the reduction, (2) Φ preserves normal forms and (3) Φ is injective on normal forms. Properties (2) and (3) are easy to check. Let us verify that Φ is a morphism:

- **resume** (**make-coroutine** $t\ C_\alpha$) **with** $x \mapsto u$
  $\equiv$ **match** (t, C_α) **with** $(x,k) \mapsto$ **abort** $(k\ u)$
  $\rightsquigarrow$ **abort** $(C_\alpha\ u\{t/x\})$
  $\equiv$ **abort** $(\lambda z.$**set-context** $\alpha\ C[z]\ u\{t/x\})$
  $\rightsquigarrow$ **abort** (**set-context** $\alpha\ C[u\{t/x\}]$)
  $\equiv$ **set-context** ε **set-context** $\alpha\ C[u\{t/x\}]$
  $\rightsquigarrow$ **set-context** $\alpha\ C[u\{t/x\}]$.
- $C[$**resume** t **with** $x \mapsto u]$
  $\equiv C[$**match** t **with** $(x,k) \mapsto$ **abort** $(k\ u)]$
  $\rightsquigarrow$ **match** t **with** $(x,k) \mapsto C[$**abort** $(k\ u)]$
  $\equiv$ **match** t **with** $(x,k) \mapsto C[$**set-context** $\varepsilon\ (k\ u)]$
  $\rightsquigarrow$ **match** t **with** $(x,k) \mapsto$ **set-context** $\varepsilon\ (k\ u)$
  $\equiv$ **resume** t **with** $x \mapsto u$.

The remaining rules are straightforward to deal with. ∎

B SND$_{\rightarrow\vee\wedge-}$ is sound and complete for SL

In this appendix, we show that SND$_{\rightarrow\vee\wedge-}$ is conservative over SND$^1_{\rightarrow\vee\wedge-}$ and thus it is sound and complete for Subtractive Logic (SL). We shall prove that any derivation of a sequent in SND$_{\rightarrow\vee\wedge-}$ can be translated into a derivation which does not contain any (right-hand side) introduction rule of implication nor any left-hand side introduction rule of subtraction, but which depends only on axioms valid in SND$^1_{\rightarrow\vee\wedge-}$.

The tricky part of the proof consists in showing that the constructive introduction rule of implication *commutes* with any other rule. Eventually, we conclude by applying the duality: the constructive left-hand side introduction rule for subtraction also *commutes* with any other rule (by duality). In order to prove this property, we have first to generalize these rules :

- We denote by hyp(Δ) the set of (occurrences of) hypotheses linked to at least one conclusion of Δ and let us define the following generalization of the constructive introduction rule of implication:

$$\frac{\Gamma \vdash \Delta}{\Gamma\backslash\{H\} \vdash \Delta\backslash S, H \rightarrow S^\vee} \quad \text{where } H \notin \text{hyp}(\Delta\backslash S)$$

  Note that H does not need to occur in Γ, and occurrences of hypotheses of S do not need to occur (and possibly none does) in Δ.
- Its dual rule, which generalizes the constructive left-hand side introduction rule for subtraction is the following (where cncl(Γ) denotes the set of occurrences of conclusions linked to at least one hypothesis of Γ)

$$\frac{\Gamma \vdash \Delta}{\Gamma\backslash S, S^\wedge - C \vdash \Delta\backslash\{C\}} \quad \text{where } C \notin \text{cncl}(\Gamma\backslash S)$$

  Again, C does not need to occur in Γ, and occurrences of hypotheses of S do not need to occur (and possibly none does) in Γ.

THEOREM B.1
The system SND $_{\rightarrow\vee\wedge-}$ is conservative over SND $^{1}_{\rightarrow\vee\wedge-}$.

PROOF. We first deal with the case of axioms (in Section B.1). Then, the main part of the proof consists in showing that the generalized introduction rule of implication *commutes* with any other rule (in Section B.2). Eventually, we conclude by applying the duality: the generalized left-hand side introduction rule for subtraction also *commutes* with any other rule (by duality). Note that this procedure terminates since the generalized rules are always applied to smaller proofs after a replacement. ∎

B.1 Axioms

PROPOSITION B.2
In SND $_{\rightarrow\vee\wedge}$, the set of annotated sequents that belong to one of the three following collections:

- $\Gamma, A \vdash \Delta, B$ where $A \vdash B$ is valid in SL, and there is at most one link which annotates this sequent, and this link binds A and B together;
- $\Gamma, Y \vdash \Delta$ where $Y \vdash \bot$ is valid in SL;
- $\Gamma \vdash \Delta, X$ where $\top \vdash X$ is valid in SL;

is closed under the generalized (right-hand side) introduction rule of implication (resp. left-hand side introduction rule for subtraction).

PROOF. Let us consider the generalized introduction rule of implication for the three collections of sequents:

1. The upper sequent has the form $\Gamma, A \vdash \Delta, B$ where $A \vdash B$ is valid in SL, and the unique link which annotates this sequent binds A and B together.
   - First case, $H \neq A$ and $B \notin S$.
   $$\frac{\Gamma, A \vdash \Delta, B}{\Gamma\backslash\{H\}, A \vdash \Delta\backslash S, B, H \rightarrow S^{\vee}}$$
   the lower sequent is indeed of the first form.
   - Second case, $H \neq A$ and H is discharged onto $S \cup \{B\}$
   $$\frac{\Gamma, A \vdash \Delta, B}{\Gamma\backslash\{H\}, A \vdash \Delta\backslash S, H \rightarrow (S^{\vee} \vee B)}$$
   the lower sequent is indeed of the third form since if $A \vdash B$ is valid in SL and $B \in S$ then $A \vdash H \rightarrow (S^{\vee} \vee B)$ is also valid.
   - Third case, $H = A$ and A is discharged onto $S \cup \{B\}$
   $$\frac{\Gamma, A \vdash \Delta, B}{\Gamma \vdash \Delta\backslash S, A \rightarrow (S^{\vee} \vee B)}$$
   the lower sequent is indeed of the third form since if $A \vdash B$ is valid in SL then $\top \vdash A \rightarrow (S^{\vee} \vee B)$ is also valid.

   In the case $H = A$ and $B \notin S$, i.e. where A is discharged onto another conclusion that B, the constructive constraint does not hold. Consequently this case has not to be considered.
2. The upper sequent has the form $\Gamma, Y \vdash \Delta$ where $Y \vdash \bot$ is valid in SL.
   - First case, $H \neq Y$ and $H \notin \text{hyp}(\Delta\backslash S)$
   $$\frac{\Gamma, Y \vdash \Delta}{\Gamma\backslash\{H\}, Y \vdash \Delta\backslash S, H \rightarrow S^{\vee}}$$
   the lower sequent is still of the second form.
   - Second case, $H = Y$ and $Y \notin \text{hyp}(\Delta\backslash S)$
   $$\frac{\Gamma, Y \vdash \Delta}{\Gamma \vdash \Delta\backslash S, Y \rightarrow S^{\vee}}$$
   Since $Y \vdash \bot$ is valid in SL, we infer that $Y \vdash S^{\vee}$ and thus $\top \vdash Y \rightarrow S^{\vee}$ is also valid in SL, and the lower sequent is thus of the third form.
3. The upper sequent has the form $\Gamma \vdash \Delta, X$ where $\top \vdash X$ is valid in SL.

- First case, $X \notin S$

$$\frac{\Gamma \vdash \Delta, X}{\Gamma \backslash \{H\} \vdash \Delta \backslash S, X, H \to S^{\vee}}$$

  the lower sequent is still of the third form.
- Second case, H is discharged onto $S \cup \{X\}$

$$\frac{\Gamma \vdash \Delta, X}{\Gamma \backslash \{H\} \vdash \Delta \backslash S, H \to (S^{\vee} \vee X)}$$

  Since $\top \vdash X$ is derivable in SL, we infer that $H \vdash S^{\vee} \vee X$ and thus $\top \vdash H \to (S^{\vee} \vee X)$ are also valid in SL, and the lower sequent is thus still of the third form.

The closure under the left-hand side introduction rule for subtraction is obtained by duality. ∎

B.2 Rules

Left-hand side weakening rule

- First case, $H \neq A$:

$$\frac{\dfrac{\Gamma \vdash^{1} \Delta}{\Gamma, A \vdash^{2} \Delta}}{\Gamma \backslash \{H\}, A \vdash^{3} \Delta \backslash S, A \to S^{\vee}}$$

  where $H \notin \text{hyp}_2(\Delta \backslash S)$ and thus $H \notin \text{hyp}_1(\Delta \backslash S)$. Replace with:

$$\frac{\dfrac{\Gamma \vdash \Delta}{\Gamma \backslash \{H\} \vdash \Delta \backslash S, A \to S^{\vee}}}{\Gamma \backslash \{H\}, A \vdash \Delta \backslash S, A \to S^{\vee}}$$

- Second case, $H = A$:

$$\frac{\dfrac{\Gamma \vdash^{1} \Delta}{\Gamma, A \vdash^{2} \Delta}}{\Gamma \vdash^{3} \Delta \backslash S, A \to S^{\vee}}$$

  where $A \notin \text{hyp}_2(\Delta \backslash S)$ and thus $A \notin \text{hyp}_1(\Delta \backslash S)$. Replace with:

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta \backslash S, A \to S^{\vee}}$$

Left-hand side contraction rule

- First case, $H = A^{z}$:

$$\frac{\dfrac{\Gamma, A^{x}, A^{y} \vdash^{1} \Delta}{\Gamma, A^{z} \vdash^{2} \Delta}}{\Gamma \vdash^{3} \Delta \backslash S, A \to S^{\vee}}$$

  where $A^{z} \notin \text{hyp}_2(\Delta \backslash S)$ and thus $A^{x} \notin \text{hyp}_1(\Delta \backslash S)$ and $A^{y} \notin \text{hyp}_1(\Delta \backslash \{B^{S}\})$. Replace with:

$$\frac{\dfrac{\Gamma, A^{x}, A^{y} \vdash \Delta}{\Gamma, A^{z} \vdash \Delta \backslash S, A \to S^{\vee}}}{\Gamma \vdash \Delta \backslash S, A \to (A \to S^{\vee})}$$

  and then cut with the sequent $A \to (A \to S^{\vee}) \vdash A \to S^{\vee}$ valid in SL.
- Second case, $H \neq A^{z}$:

$$\frac{\dfrac{\Gamma, A^{x}, A^{y} \vdash^{1} \Delta}{\Gamma, A^{z} \vdash^{2} \Delta}}{\Gamma \backslash \{H\}, A^{z} \vdash^{3} \Delta \backslash S, H \to S^{\vee}}$$

  where $H \notin \text{hyp}_2(\Delta \backslash S)$ and thus $H \notin \text{hyp}_1(\Delta \backslash S)$. Replace with:

$$\frac{\dfrac{\Gamma, A^{x}, A^{y}, H \vdash \Delta}{\Gamma \backslash \{H\}, A^{x}, A^{y} \vdash \Delta \backslash S, H \to S^{\vee}}}{\Gamma \backslash \{H\}, A^{z} \vdash \Delta \backslash S, H \to S^{\vee}}$$

Right-hand side contraction rule

- First case, $B \notin S$:

$$\dfrac{\dfrac{\Gamma \vdash^1 \Delta, B^\alpha, B^\beta}{\Gamma \vdash^2 \Delta, B^\gamma}}{\Gamma\backslash\{H\} \vdash^3 \Delta\backslash S, H \to S^\vee, B^\gamma}$$

  where $H \notin \mathrm{hyp}_2(\Delta\backslash S, B^\gamma)$ and thus $H \notin \mathrm{hyp}_1(\Delta\backslash S, B^\alpha, B^\beta)$. Replace with:

$$\dfrac{\dfrac{\Gamma \vdash \Delta, B, B}{\Gamma\backslash\{H\} \vdash \Delta\backslash S, H \to S^\vee, B, B}}{\Gamma\backslash\{H\} \vdash \Delta\backslash S, H \to S^\vee, B}$$

- Second case, H is discharged onto $S \cup \{B\}$:

$$\dfrac{\dfrac{\Gamma \vdash^1 \Delta, B^\alpha, B^\beta}{\Gamma \vdash^2 \Delta, B^\gamma}}{\Gamma\backslash\{H\} \vdash^3 \Delta\backslash S, H \to (S^\vee \vee B)}$$

  where $H \notin \mathrm{hyp}_2(\Delta\backslash S, B^\gamma)$ and thus $H \notin \mathrm{hyp}_1(\Delta\backslash S, B^\alpha, B^\beta)$. Replace with:

$$\dfrac{\Gamma \vdash \Delta, B, B}{\Gamma\backslash\{H\} \vdash \Delta\backslash S, H \to (S^\vee \vee B \vee B)}$$

  and then cut with the sequent $H \to (S^\vee \vee B \vee B) \vdash H \to (S^\vee \vee B)$ valid in SL.

Right-hand side weakening rule

- First case, $B \notin S$:

$$\dfrac{\dfrac{\Gamma \vdash^1 \Delta}{\Gamma \vdash^2 \Delta, B}}{\Gamma\backslash\{H\} \vdash^3 \Delta\backslash S, H \to S^\vee, B}$$

  where $H \notin \mathrm{hyp}_2(\Delta\backslash S, B)$ and thus $H \notin \mathrm{hyp}_1(\Delta\backslash S)$. Replace with:

$$\dfrac{\dfrac{\Gamma \vdash \Delta}{\Gamma\backslash\{H\} \vdash \Delta\backslash S, H \to S^\vee}}{\Gamma\backslash\{H\} \vdash \Delta\backslash S, H \to S^\vee, B}$$

- Second case, H is discharged onto $S \cup \{B\}$:

$$\dfrac{\dfrac{\Gamma \vdash^1 \Delta}{\Gamma \vdash^2 \Delta, B^\alpha}}{\Gamma\backslash\{H\} \vdash^3 \Delta\backslash S, H \to (S^\vee \vee B)}$$

  where $H \notin \mathrm{hyp}_2(\Delta\backslash S, B^\alpha)$ and thus $H \notin \mathrm{hyp}_1(\Delta\backslash S)$. Replace with:

$$\dfrac{\Gamma \vdash \Delta}{\Gamma\backslash\{H\} \vdash \Delta\backslash S, H \to S^\vee}$$

  and then cut with the sequent $H \to S^\vee \vdash H \to (S^\vee \vee B)$ valid in SL.

Cut rule

$$\dfrac{\dfrac{\Gamma' \vdash^1 A, \Delta' \quad \Gamma'', A \vdash^4 \Delta''}{\Gamma', \Gamma'' \vdash^2 \Delta', \Delta''}}{(\Gamma', \Gamma'')\backslash\{H\} \vdash^3 (\Delta', \Delta'')\backslash S, H \to S^\vee}$$

where $H \notin \mathrm{hyp}_2((\Delta', \Delta'')\backslash S)$ and thus, by setting $S' = S \cap \Gamma'$ and $S'' = S \cap \Gamma''$:

- First case, $H \notin \Gamma'$. Replace with:

$$\dfrac{\dfrac{\Gamma' \vdash \Delta', A}{\Gamma' \vdash \Delta' \backslash S', H \to S'^{\vee}, A} \quad \dfrac{\Gamma'', A \vdash \Delta''}{\Gamma'' \backslash \{H\}, A \vdash \Delta'' \backslash S', H \to S''^{\vee}}}{\dfrac{(\Gamma', \Gamma'') \backslash \{H\} \vdash (\Delta', \Delta'') \backslash S, H \to S'^{\vee}, H \to S''^{\vee}}{(\Gamma', \Gamma'') \backslash \{H\} \vdash (\Delta', \Delta'') \backslash S, (H \to S'^{\vee}) \vee (H \to S''^{\vee})}}$$

  then cut with the sequent $(H \to S'^{\vee}) \vee (H \to S''^{\vee}) \vdash H \to (S'^{\vee} \vee S''^{\vee})$ valid in SL.
- Second case, $H \notin \Gamma''$ and $H \notin \mathrm{hyp}_1(A)$ since $H \notin \mathrm{hyp}_1(A, \Delta' \backslash S')$. Replace with:

$$\dfrac{\dfrac{\Gamma' \vdash \Delta', A}{\Gamma' \backslash \{H\} \vdash \Delta' \backslash S', H \to S'^{\vee}, A} \quad \dfrac{\Gamma'', A \vdash \Delta''}{\Gamma'', A \vdash \Delta'' \backslash S', H \to S''^{\vee}}}{\dfrac{(\Gamma', \Gamma'') \backslash \{H\} \vdash (\Delta', \Delta'') \backslash S, H \to S'^{\vee}, H \to S''^{\vee}}{(\Gamma', \Gamma'') \backslash \{H\} \vdash (\Delta', \Delta'') \backslash S, (H \to S'^{\vee}) \vee (H \to S''^{\vee})}}$$

  then cut with the sequent $(H \to S'^{\vee}) \vee (H \to S''^{\vee}) \vdash H \to (S'^{\vee} \vee S''^{\vee})$ valid in SL.
- Third case, $H \in \Gamma'$ and $H \in \mathrm{hyp}_1(A)$ and since $H \notin \mathrm{hyp}_1(\Delta'' \backslash S'')$ we have $A \notin \mathrm{hyp}_1(\Delta'' \backslash S')$. Replace with:

$$\dfrac{\dfrac{\Gamma' \vdash \Delta', A}{\Gamma' \backslash \{H\} \vdash \Delta' \backslash S', H \to (S'^{\vee} \vee A)} \quad \dfrac{\Gamma'', A \vdash \Delta''}{\Gamma'' \vdash \Delta' \backslash S', A \to S''^{\vee}}}{(\Gamma', \Gamma'') \backslash \{H\} \vdash (\Delta', \Delta'') \backslash S, (H \to (S'^{\vee} \vee A)) \wedge (A \to S''^{\vee})}$$

  then cut with the sequent $(H \to (S'^{\vee} \vee A)) \wedge (A \to S''^{\vee}) \vdash H \to (S'^{\vee} \vee S''^{\vee})$ valid in SL.

Right-hand side elimination rule for the implication

- First case, $B \notin S$:

$$\dfrac{\dfrac{\Gamma' \vdash^1 \Delta', A \to B \quad \Gamma'' \vdash^4 \Delta'', A}{\Gamma', \Gamma'' \vdash^2 \Delta', \Delta'', B}}{(\Gamma', \Gamma'') \backslash \{H\} \vdash^3 (\Delta', \Delta'') \backslash S, B, H \to S^{\vee}}$$

  where $H \notin \mathrm{hyp}_2(B, (\Delta', \Delta'') \backslash S)$ and thus, by setting $S' = S \cap \Gamma'$ and $S'' = S \cap \Gamma''$, we have $H \notin \mathrm{hyp}_1(A \to B, \Delta \backslash S')$ and $H \notin \mathrm{hyp}_4(A, \Delta \backslash S')$. Replace with:

$$\dfrac{\dfrac{\Gamma' \vdash \Delta', A \to B}{\Gamma' \backslash \{H\} \vdash \Delta' \backslash S', H \to S'^{\vee}, A \to B} \quad \dfrac{\Gamma'' \vdash \Delta'', A}{\Gamma'' \backslash \{H\} \vdash \Delta'' \backslash S'', H \to S''^{\vee}, A}}{\dfrac{(\Gamma', \Gamma'') \backslash \{H\} \vdash (\Delta', \Delta'') \backslash S, H \to S'^{\vee}, H \to S''^{\vee}, B}{(\Gamma', \Gamma'') \backslash \{H\} \vdash (\Delta', \Delta'') \backslash S, (H \to S'^{\vee}) \vee (H \to S''^{\vee}), B}}$$

  and then cut with the sequent $(H \to S'^{\vee}) \vee (H \to S''^{\vee}) \vdash H \to (S'^{\vee} \vee S''^{\vee})$ valid in SL.
- Second case, H is discharged onto $S \cup \{B\}$:

$$\dfrac{\dfrac{\Gamma' \vdash^1 \Delta', A \to B \quad \Gamma'' \vdash^4 \Delta'', A}{\Gamma', \Gamma'' \vdash^2 \Delta', \Delta'', B}}{(\Gamma', \Gamma'') \backslash \{H\} \vdash^3 (\Delta', \Delta'') \backslash S, H \to (S^{\vee} \vee B)}$$

  where $H \notin \mathrm{hyp}_2((\Delta', \Delta'') \backslash S)$ and thus, by setting $S' = S \cap \Gamma'$ and $S'' = S \cap \Gamma''$, there is $H \notin \mathrm{hyp}_1(\Delta \backslash S')$ and $H \notin \mathrm{hyp}_4(\Delta \backslash S')$. Replace with:

$$\dfrac{\dfrac{\Gamma' \vdash \Delta', A \to B}{\Gamma' \backslash \{H\} \vdash \Delta' \backslash S', H \to (S'^{\vee} \vee (A \to B))} \quad \dfrac{\Gamma'' \vdash \Delta'', A}{\Delta'' \backslash S', H \to (S''^{\vee} \vee A)}}{(\Gamma', \Gamma'') \backslash \{H\} \vdash (\Delta', \Delta'') \backslash S, (H \to (S'^{\vee} \vee (A \to B))) \wedge (H \to (S''^{\vee} \vee A))}$$

  and then cut with the sequent $(H \to (S'^{\vee} \vee (A \to B))) \wedge (H \to (S''^{\vee} \vee A)) \vdash H \to (S^{\vee} \vee B)$ valid in SL.

Left-hand side introduction rule of disjunction

- First case, $H \neq A \vee B$:

$$\dfrac{\dfrac{\Gamma', A \vdash^1 \Delta' \quad \Gamma'', B \vdash^4 \Delta''}{\Gamma', \Gamma'', A \vee B \vdash^2 \Delta', \Delta''}}{(\Gamma', \Gamma'')\backslash\{H\}, A \vee B \vdash^3 (\Delta', \Delta'')\backslash S, H \to S^{\vee}}$$

  where $H \notin \text{hyp}_2((\Delta', \Delta'')\backslash S)$ and thus, by setting $S' = S \cap \Gamma'$ and $S'' = S \cap \Gamma''$, we have $H \notin \text{hyp}_1(\Delta\backslash S')$ and $H \notin \text{hyp}_4(\Delta\backslash S')$. Replace with:

$$\dfrac{\dfrac{\dfrac{\Gamma', A \vdash \Delta'}{\Gamma'\backslash\{H\}, A \vdash \Delta'\backslash S', H \to S'^{\vee}} \quad \dfrac{\Gamma'', B \vdash \Delta''}{\Gamma''\backslash\{H\}, B \vdash \Delta'', H \to S''^{\vee}}}{(\Gamma', \Gamma'')\backslash\{H\}, A \vee B \vdash (\Delta', \Delta'')\backslash S, H \to S'^{\vee}, H \to S''^{\vee}}}{(\Gamma', \Gamma'')\backslash\{H\}, A \vee B \vdash (\Delta', \Delta'')\backslash S, (H \to S'^{\vee}) \vee (H \to S''^{\vee})}$$

  and then cut with the sequent $(H \to S'^{\vee}) \vee (H \to S''^{\vee}) \vdash H \to (S'^{\vee} \vee S''^{\vee})$ valid in SL.

- Second case, $H = A \vee B$:

$$\dfrac{\dfrac{\Gamma', A \vdash^1 \Delta' \quad \Gamma'', B \vdash^4 \Delta''}{\Gamma', \Gamma'', A \vee B \vdash^2 \Delta', \Delta''}}{\Gamma', \Gamma'' \vdash^3 (\Delta', \Delta'')\backslash S, A \vee B \to S^{\vee}}$$

  where $A \vee B \notin \text{hyp}_2((\Delta', \Delta'')\backslash S)$ and thus, by setting $S' = S \cap \Gamma'$ and $S'' = S \cap \Gamma''$, we have $A \notin \text{hyp}_1(\Delta\backslash S')$ and $B \notin \text{hyp}_4(\Delta\backslash S')$. Replace with:

$$\dfrac{\dfrac{\Gamma', A \vdash \Delta'}{\Gamma'\backslash\{H\} \vdash \Delta'\backslash S', A \to S'^{\vee}} \quad \dfrac{\Gamma'', B \vdash \Delta''}{\Delta''\backslash S', B \to S''^{\vee}}}{(\Gamma', \Gamma'')\backslash\{H\} \vdash (\Delta', \Delta'')\backslash S, (A \to S'^{\vee}) \wedge (B \to S''^{\vee})}$$

  and then cut with the sequent $(A \to S'^{\vee}) \wedge (B \to S''^{\vee}) \vdash (A \vee B) \to (S'^{\vee} \vee S''^{\vee})$ valid in SL.

Left-hand side elimination rule of disjunction

We consider only the case of the first *injection*, the second one is similar.

- First case, $H \neq A$:

$$\dfrac{\dfrac{\Gamma, A \vee B \vdash^1 \Delta}{\Gamma, A \vdash^2 \Delta}}{\Gamma\backslash\{H\}, A \vdash^3 \Delta\backslash S, H \to S^{\vee}}$$

  where $H \notin \text{hyp}_2(\Delta\backslash S)$ and thus $H \notin \text{hyp}_1(\Delta\backslash S)$. Replace with:

$$\dfrac{\dfrac{\Gamma, A \vee B \vdash \Delta}{\Gamma\backslash\{H\}, A \vee B \vdash \Delta\backslash S, H \to S^{\vee}}}{\Gamma\backslash\{H\}, A \vdash \Delta\backslash S, H \to S^{\vee}}$$

- Second case, $H = A$:

$$\dfrac{\dfrac{\Gamma, A \vee B \vdash^1 \Delta}{\Gamma, A \vdash^2 \Delta}}{\Gamma\backslash\{H\} \vdash^3 \Delta\backslash S, A \to S^{\vee}}$$

  where $H \notin \text{hyp}_2(\Delta\backslash S)$ and thus $H \notin \text{hyp}_1(\Delta\backslash S)$ and $A \notin \text{hyp}_4(\Delta\backslash S)$. Replace with:

$$\dfrac{\Gamma, A \vee B \vdash \Delta}{\Gamma\backslash\{H\} \vdash \Delta\backslash S, (A \vee B) \to S^{\vee}}$$

  and then cut with the sequent $(A \vee B) \to S^{\vee} \vdash A \to S^{\vee}$ valid in SL.

Introduction rule for conjunction

- First case, $A \wedge B \notin S$:

$$\dfrac{\dfrac{\Gamma' \vdash^1 \Delta', A \quad \Gamma'' \vdash^4 \Delta'', B}{\Gamma', \Gamma'' \vdash^2 \Delta', \Delta'', A \wedge B}}{(\Gamma', \Gamma'')\backslash\{H\} \vdash^3 (\Delta', \Delta'')\backslash S, A \wedge B, H \to S^\vee}$$

where $H \notin \text{hyp}_2(A, B, (\Delta', \Delta'')\backslash S)$ and thus by setting $S' = S \cap \Gamma'$ et $S'' = S \cap \Gamma''$, we have $H \notin \text{hyp}_1(A, \Delta\backslash S')$ and $H \notin \text{hyp}_4(B, \Delta\backslash S')$. Replace with:

$$\dfrac{\dfrac{\dfrac{\Gamma' \vdash \Delta', A}{\Gamma'\backslash\{H\} \vdash \Delta'\backslash S', H \to S'^\vee, A} \quad \dfrac{\Gamma'' \vdash \Delta'', B}{\Gamma''\backslash\{H\} \vdash \Delta''\backslash S'', H \to S''^\vee, B}}{(\Gamma', \Gamma'')\backslash\{H\} \vdash (\Delta', \Delta'')\backslash S, H \to S'^\vee, H \to S''^\vee, A \wedge B}}{(\Gamma', \Gamma'')\backslash\{H\} \vdash (\Delta', \Delta'')\backslash S, (H \to S'^\vee) \vee (H \to S''^\vee), A \wedge B}$$

then cut with the sequent $(H \to S'^\vee) \vee (H \to S''^\vee) \vdash H \to (S'^\vee \vee S''^\vee)$ valid in SL.

- Second case, H is discharged onto $S \cup \{A \wedge B\}$:

$$\dfrac{\dfrac{\Gamma' \vdash^1 \Delta', A \quad \Gamma'' \vdash^4 \Delta'', B}{\Gamma', \Gamma'' \vdash^2 \Delta', \Delta'', A \wedge B}}{(\Gamma', \Gamma'')\backslash\{H\} \vdash^3 (\Delta', \Delta'')\backslash S, H \to (S^\vee \vee (A \wedge B))}$$

where $H \notin \text{hyp}_2((\Delta', \Delta'')\backslash S)$ and thus by setting $S' = S \cap \Gamma'$ et $S'' = S \cap \Gamma''$, on a $H \notin \text{hyp}_1(\Delta\backslash S')$ and $H \notin \text{hyp}_4(\Delta\backslash S')$. Replace with:

$$\dfrac{\dfrac{\dfrac{\Gamma' \vdash \Delta', A}{\Gamma'\backslash\{H\} \vdash \Delta'\backslash S', H \to (S'^\vee \vee A)} \quad \dfrac{\Gamma'' \vdash \Delta'', B}{\Delta''\backslash S', H \to (S''^\vee \vee B)}}{(\Gamma', \Gamma'')\backslash\{H\} \vdash (\Delta', \Delta'')\backslash S, (H \to (S'^\vee \vee A)) \wedge (H \to (S''^\vee \vee B))}}{}$$

then cut with the sequent $(H \to (S'^\vee \vee A)) \wedge (H \to (S''^\vee \vee B)) \vdash H \to (S^\vee \vee (A \wedge B))$ valid in SL.

Elimination rule for conjunction

We deal only with the first projection, the second one is similar.

- First case, $A \notin S$:

$$\dfrac{\dfrac{\Gamma \vdash^1 \Delta, A \wedge B}{\Gamma \vdash^2 \Delta, A}}{\Gamma\backslash\{H\} \vdash^3 \Delta\backslash S, H \to S^\vee, A}$$

where $H \notin \text{hyp}_2(A, \Delta\backslash S)$ and thus $H \notin \text{hyp}_1(A \wedge B, \Delta\backslash S)$. Replace with:

$$\dfrac{\dfrac{\Gamma \vdash \Delta, A \wedge B}{\Gamma\backslash\{H\} \vdash \Delta\backslash S, H \to S^\vee, A \wedge B}}{\Gamma\backslash\{H\} \vdash \Delta\backslash S, H \to S^\vee, A}$$

- Second case, H is discharged onto $S \cup \{A\}$:

$$\dfrac{\dfrac{\Gamma \vdash^1 \Delta, A \wedge B}{\Gamma \vdash^2 \Delta, A}}{\Gamma\backslash\{H\} \vdash^3 \Delta\backslash S, H \to (S^\vee \vee A)}$$

where $H \notin \text{hyp}_2(\Delta\backslash S)$ and thus $H \notin \text{hyp}_1(\Delta\backslash S)$ and $A \notin \text{hyp}_4(\Delta\backslash S)$. Replace with:

$$\dfrac{\Gamma \vdash \Delta, A \wedge B}{\Gamma\backslash\{H\} \vdash \Delta\backslash S, H \to (S^\vee \vee (A \wedge B))}$$

then cut with the sequent $H \to (S^\vee \vee (A \wedge B)) \vdash H \to (S^\vee \vee A)$ valid in SL.

Left-hand side elimination rule for subtraction

- First case, $H \neq A$:

$$\dfrac{\dfrac{\Gamma', A - B \vdash^1 \Delta' \quad \Gamma'', B \vdash^4 \Delta''}{\Gamma', \Gamma'', A \vdash^2 \Delta', \Delta''}}{(\Gamma', \Gamma'')\backslash\{H\}, A \vdash^3 (\Delta', \Delta'')\backslash S, H \to S^\vee}$$

where $H \notin \text{hyp}_2((\Delta', \Delta'')\backslash S)$ and thus by setting $S' = S \cap \Gamma'$ et $S'' = S \cap \Gamma''$, we have $H \notin \text{hyp}_1(\Delta\backslash S')$ and $H \notin \text{hyp}_4(\Delta\backslash S')$. Replace with:

$$\dfrac{\dfrac{\dfrac{\Gamma', A - B \vdash \Delta'}{\Gamma'\backslash\{H\}, A - B \vdash \Delta'\backslash S', H \to S'^\vee} \quad \dfrac{\Gamma'', B \vdash \Delta''}{\Gamma''\backslash\{H\}, B \vdash \Delta'', H \to S''^\vee}}{(\Gamma', \Gamma'')\backslash\{H\}, A \vdash (\Delta', \Delta'')\backslash S, H \to S'^\vee, H \to S''^\vee}}{(\Gamma', \Gamma'')\backslash\{H\}, A \vdash (\Delta', \Delta'')\backslash S, (H \to S'^\vee) \vee (H \to S''^\vee)}$$

then cut with the sequent $(H \to S'^\vee) \vee (H \to S''^\vee) \vdash H \to (S'^\vee \vee S''^\vee)$ valid in SL.

- Second case, $H = A$:

$$\dfrac{\dfrac{\Gamma', A - B \vdash^1 \Delta' \quad \Gamma'', B \vdash^4 \Delta''}{\Gamma', \Gamma'', A \vdash^2 \Delta', \Delta''}}{\Gamma', \Gamma'' \vdash^3 (\Delta', \Delta'')\backslash S, A \to S^\vee}$$

where $A \notin \text{hyp}_2((\Delta', \Delta'')\backslash S)$ and thus by setting $S' = S \cap \Gamma'$ and $S'' = S \cap \Gamma''$, we have $A - B \notin \text{hyp}_1(\Delta\backslash S')$ and $B \notin \text{hyp}_4(\Delta\backslash S')$. Replace with:

$$\dfrac{\dfrac{\dfrac{\Gamma', A - B \vdash \Delta'}{\Gamma'\backslash\{H\} \vdash \Delta'\backslash S', (A - B) \to S'^\vee} \quad \dfrac{\Gamma'', B \vdash \Delta''}{\Delta''\backslash S', B \to S''^\vee}}{(\Gamma', \Gamma'')\backslash\{H\} \vdash (\Delta', \Delta'')\backslash S, ((A - B) \to S'^\vee) \wedge (B \to S''^\vee)}}{}$$

then cut with the sequent $((A - B) \to S'^\vee) \wedge (B \to S''^\vee) \vdash (A \to (S'^\vee \vee S''^\vee)$ valid in SL.

Left-hand side introduction rule for subtraction

- First case, $H \neq A - B$:

$$\dfrac{\dfrac{\Gamma, A \vdash^1 \Delta, B}{\Gamma, A - B \vdash^2 \Delta}}{\Gamma\backslash\{H\}, A - B \vdash^3 \Delta\backslash S, H \to S^\vee}$$

where $H \notin \text{hyp}_2(\Delta\backslash S)$ and $H \notin \text{hyp}_1(B)$ and thus $H \notin \text{hyp}_1(\Delta\backslash S, B)$. Replace with:

$$\dfrac{\dfrac{\Gamma, A \vdash \Delta, B}{\Gamma\backslash\{H\}, A \vdash \Delta\backslash S, H \to S^\vee, B}}{\Gamma\backslash\{H\}, A - B \vdash \Delta\backslash S, H \to S^\vee}$$

- Second case, $H = A - B$:

$$\dfrac{\dfrac{\Gamma, A \vdash^1 \Delta, B}{\Gamma, A - B \vdash^2 \Delta}}{\Gamma \vdash^3 \Delta\backslash S, (A - B) \to S^\vee}$$

where $H \notin \text{hyp}_2(\Delta\backslash S)$ and $H \notin \text{hyp}_1(B)$ and thus $H \notin \text{hyp}_1(\Delta\backslash S, B)$. Replace with:

$$\dfrac{\Gamma, A \vdash \Delta, B}{\Gamma \vdash \Delta\backslash S, A \to (S^\vee \vee B)}$$

then cut with the sequent $A \to (S^\vee \vee B) \vdash (A - B) \to S^\vee$ valid in SL.

Received 28 February 2003