

# What are acceptable reductions?

Perspectives from proof-theoretic semantics and type theory

Sara Ayhan\*

Ruhr University Bochum, Institute of Philosophy I  
sara.ayhan@rub.de

## Abstract

It has been argued that reduction procedures are closely connected to the question about identity of proofs and that accepting certain reductions would lead to a trivialization of identity of proofs in the sense that every derivation of the same conclusion would have to be identified. In this paper it will be shown that the question, which reductions we accept in our system, is not only important if we see them as generating a theory of proof identity but is also decisive for the more general question whether a proof has meaningful content. There are certain reductions which would not only force us to identify proofs of *different arbitrary formulas* but which would render derivations in a system allowing them *meaningless*. To exclude such cases, a minimal criterion is proposed which reductions have to fulfill to be acceptable.

**Keywords:** Reductions, Proof-theoretic semantics, Identity of proofs, Sense, Curry-Howard

## 1 Introduction

What are acceptable reductions<sup>1</sup> in the context of proofs and why is it important to distinguish these from ‘bad’ ones? As Schroeder-Heister and Tranchini (2017, p. 574) argue, from a philosophical point of view, or more specifically a standpoint of proof-theoretic semantics (see Schroeder-Heister, 2022), reduction procedures are closely connected to the question about identity of proofs: If we take proofs to be abstract entities represented by (natural deduction) derivations, then derivations belonging to the same equivalence class induced by the reflexive, symmetric, and transitive closure of reducibility can be said to represent the same proof object.<sup>2</sup> As they show, accepting certain reductions, more specifically accepting the so-called *Ekman-reduction* (see below), would lead to a trivialization of identity of proofs in the sense that every derivation of the same conclusion would have to be identified. They suggest such a trivialization as a criterion to disallow reductions. I will argue that the question, which reductions we accept in our system, is not only important

---

\*I would like to thank David Ripley, Ansten Klev, Will Stafford and Luca Tranchini for reading early versions of this paper and giving me extensive feedback on it, which immensely helped me to clarify my ideas. Thanks also go to an anonymous referee for their very constructive and helpful report. I am especially grateful to Heinrich Wansing for his thorough reading and feedback on the paper and his kind and valuable suggestions for improvement.

<sup>1</sup>Note that I am focussing strictly on *reductions* in this paper, i.e., procedures that cut out what is in some way considered a detour of a derivation, not conversions in general, like expansions or permutations. The latter are certainly also of great interest for proof-theoretic semantics but would extend the scope of this paper.

<sup>2</sup>This goes back to Prawitz (1971, pp. 257-261), who credits the idea to Per Martin-Löf; many others have defended this view since.

if we see them as generating a theory of proof identity but is also decisive for the more general question whether a proof has meaningful content, i.e., it does not only matter to the question about the *denotation* of proofs but also to the question about their *sense*. Therefore, we need to be careful: We cannot just accept any reduction, i.e., any procedure eliminating some kind of detour in a derivation.

An example of a reduction not belonging to the usual reductions is Ekman-reduction, as it is presented in (Ekman, 1994, 1998) and extensively discussed by Schroeder-Heister and Tranchini (2017; 2018) and Tennant (2021):

$$\frac{\frac{B \rightarrow A \quad \frac{A \rightarrow B \quad A}{B} \rightarrow E}{A} \rightarrow E}{A} \rightsquigarrow \frac{\mathcal{D}}{A}$$

What we want in light of such a non-standard kind of reduction are criteria determining which reductions can be allowed in our system and which should be dismissed. It is advantageous for such an approach to exploit the so-called *Curry-Howard correspondence* (see, e.g., Sørensen & Urzyczyn, 2006) and examine proof systems annotated with  $\lambda$ -terms. These make the structure of our derivations explicit and facilitate to show what is wrong with potential reductions and why they should not be admitted in our system. The question then shifts to asking which reduction procedures for *terms* can be allowed. The  $\lambda$ -calculus and some well-known properties thereof can provide us with directions as to what could be (un)desirable features of reductions.

Annotating Ekman-reduction with terms then shows something else - besides Schroeder-Heister and Tranchini's point - that is essentially problematic about this reduction. Indeed, I want to show that allowing it would be equal to allowing a reduction for **tonk**, i.e., a reduction for a derivation consisting of a **tonk**-introduction rule followed by its elimination rule (see below, section 3.1.1). It is generally agreed upon, though, that there cannot be a sensible reduction for this connective. By creating a **tonk**-reduction in the same fashion as it would be done for other connectives, the consequences of allowing this reduction are made explicit and it can be shown that those would be the same for Ekman-reduction: Not only would these reductions induce an equivalence relation relating different terms in normal form of the *same* conclusion, they would also allow to reduce a term of one type to a term of an *arbitrary* other. If we take reductions as generating identity between proofs, then that would also force us to identify proofs of arbitrarily different formulas. But even if we reject this assumption (some researchers do not find this theory of proof identity very compelling, see section 4), I will argue that allowing such reductions would render derivations in such a system *meaningless*.

## 2 Reduction procedures in natural deduction and $\lambda$ -calculus

The reductions for the connectives of, say, minimal propositional logic, corresponding to  $\beta$ -reductions in  $\lambda$ -calculus, are meant to eliminate unnecessary detours of the following form: There is a formula, called *maximal formula*, which is both the conclusion of an application of an introduction rule of a connective as well as the major premise of an applied elimination rule governing the same connective. It can be shown for those connectives (see below for  $\rightarrow$ ) that in these cases the maximal formula (below  $A \rightarrow B$ ) can be eliminated without losing anything essential because, as Prawitz (1971, p. 251) argues, this procedure is just a way to make the *inversion principle*<sup>3</sup> explicit.

<sup>3</sup>The principle, to which (at least) the rules governing the connectives of minimal logic adhere, saying that nothing new is obtained by an elimination immediately following an introduction of the major premise of the elimination rule (Prawitz, 1971).

$$\frac{\frac{\frac{[A]}{\vdots \mathscr{D}}}{B} \rightarrow I}{A \rightarrow B} \rightarrow E}{B} \rightsquigarrow \frac{\mathscr{D}'}{\frac{[A]}{\vdots \mathscr{D}}}{B} \rightarrow E$$

It can and has been argued, however, that there are more reductions than the ones for the connectives that are usually considered (see, e.g., Tennant, 1995). One of those, presented in (Ekman, 1994, 1998), will be discussed in this paper.

As mentioned above, in using a term-annotated proof system we are implementing the Curry-Howard correspondence, which takes the view of proofs as programs and formulas as types as a basis and which states a close correspondence between these notions in the simply typed  $\lambda$ -calculus and natural deduction (ND) systems of intuitionistic logic. For our purposes here it suffices to consider the  $\rightarrow$ -fragment of intuitionistic logic and correspondingly the system  $\lambda^{\rightarrow}$ . We use  $\rho, \sigma, \tau, \dots$  for arbitrary atomic formulas,  $A, B, C, \dots$  for arbitrary formulas, and  $\Gamma, \Delta, \dots$  for sets of formulas. The concatenation  $\Gamma, A$  stands for  $\Gamma \cup \{A\}$ . For term variables,  $x, y, z, \dots$  are used and  $r, s, t, \dots$  for arbitrary terms. Furthermore, we use ‘ $\equiv$ ’ to denote *syntactic identity* between terms, types, or derivations. The following are our term-annotated ND-rules with the corresponding  $\beta$ -reduction:

$$\frac{\frac{[x : A]}{\vdots}}{t : B} \rightarrow I \quad \frac{\frac{\Gamma}{\vdots} \quad \frac{\Delta}{\vdots}}{s : A \rightarrow B \quad t : A} \rightarrow E}{App(s, t) : B} \rightarrow E \quad App(\lambda x.t, s) \rightsquigarrow_{\beta} t[s/x]$$

We read  $t : A$  as “term  $t$  is of type  $A$ ” or, in the ‘proof-reading’, “ $t$  is a proof of formula  $A$ ”. Such an expression is also called a *type assignment statement* with the  $\lambda$ -term being the *subject* and the type the *predicate*. Thus, we use a type system *à la Curry* here, in which the terms are not typed, in the sense that the types are part of the term’s structure, but are *assigned* types according to type assignment rules, which in our case are simply the rules above. With  $t : A$  we express that  $A$  is the *principal type* of term  $t$ , i.e., the most general type that can be assigned to  $t$ .<sup>4</sup> Substitution is expressed by  $t[s/x]$ , meaning that in term  $t$  every free occurrence of  $x$  is substituted with  $s$ . The usual capture-avoiding requirements for variable substitution are to be observed. I will follow standard terminology of type theory here and call a term of the form  $App(\lambda x.t, s)$  a  $\beta$ -redex and the corresponding term  $t[s/x]$  its *contractum*.<sup>5</sup> Replacing an occurrence of a  $\beta$ -redex contained in a term by its contractum is called a  $\beta$ -contraction and if there is a finite (possibly empty) series of  $\beta$ -contractions changing term  $t$  to  $t'$ , we say that  $t$   $\beta$ -reduces to  $t'$  and write  $t \rightsquigarrow_{\beta} t'$ . The reduction relation  $\rightsquigarrow_{\beta}$  is reflexive and transitive and closed under  $\alpha$ -conversion, i.e., renaming of bound variables. A term that contains no  $\beta$ -redexes is said to be  $\beta$ -normal or in  $\beta$ -normal form ( $\beta$ -nf) and  $t'$  is the  $\beta$ -nf of  $t$  if  $t \rightsquigarrow_{\beta} t'$  and  $t'$  is  $\beta$ -normal.

In general (if we consider more connectives than  $\rightarrow$ ), the correspondence to the introduction and elimination rules in  $\lambda$ -calculus is that each connective has its own *constructor*, an operator constructing canonical objects of particular types, and a *destructor* specifying the use of these objects in computations. A  $\beta$ -redex consists of a destructor applied to the constructor of the same connective, so Curry-Howard correspondence always gives us an analogy between a  $\beta$ -redex and a proof detour consisting of an elimination immediately following the introduction of the same connective (Sørensen & Urzyczyn, 2006, p. 87f.).

<sup>4</sup>For example, for the term  $\lambda x.x$ , its types could be  $p \rightarrow p$ ,  $q \rightarrow q$ ,  $(p \rightarrow q) \rightarrow (p \rightarrow q)$ , etc., while its principal type would be  $A \rightarrow A$ .

<sup>5</sup>For the following definitions (with only slightly differing formulations and notations), see (Barendregt, 1992), (Girard, 1989), and (Hindley & Seldin, 2008). I will use the same terminology (without the ‘ $\beta$ -’) for any reduction procedures, whether or not they will be found acceptable in the course of the paper. For reduction procedures in general, see also (Baader & Nipkow, 1998).

It is important to stress that reduction procedures can be interpreted in two different (yet certainly closely related) ways (see, e.g., (Barendregt, 1992, Ch. 2.3), (Girard, 1989, pp. 18-20), (Hindley & Seldin, 2008, pp. 11-18)). One interpretation is to see reductions as inducing an identity relation, i.e., on this view the relation applies equally in both directions. We will speak of  $\beta$ -equality of terms in these cases and use  $=_\beta$  to express this relation. It is just like, e.g.,  $App(\lambda x.x^2 + 7, 2) \rightsquigarrow_\beta 11$  expresses the fact that  $4+7=11$ .

Another interpretation of reductions is to see them as *directed* computations, calculations, or executions corresponding to the idea of program evaluation. On this view, the asymmetry between redex and contractum must be stressed: In our example ‘4+7’ can be interpreted as ‘11’ by doing a calculation. A reduction procedure is seen as an evaluation that is run on a term and thereby interprets this term in a different way. The non-symmetric  $\beta$ -reducibility relation implies the symmetric relation of  $\beta$ -equality but not the other way around (Hindley & Seldin, 2008, p. 16). Hence, if  $t \rightsquigarrow_\beta t'$ , then  $t =_\beta t'$ ; but not: if  $t =_\beta t'$ , then  $t \rightsquigarrow_\beta t'$ . Just like  $4+7$  evaluates to 11 but not the other way around: 11 is fully evaluated; it is already in normal form, i.e., we do not *reduce* it to  $4+7$ .

One of the most important results in  $\lambda$ -calculus, which will also be important for this paper, is the so-called *Church-Rosser Theorem* stating the *confluence property* for  $\beta$ -reduction:

**Church-Rosser Theorem:** If a term can be reduced to two syntactically different terms, then there is a term to which these two can be reduced. Put formally, if  $t \rightsquigarrow_\beta t'$  and  $t \rightsquigarrow_\beta t''$ , then there is a term  $s$  such that  $t' \rightsquigarrow_\beta s$  and  $t'' \rightsquigarrow_\beta s$ .

Likewise, this property holds for  $\beta$ -equality: if two syntactically different terms are  $\beta$ -equal, then there is a term to which they both can be reduced in finitely many steps, i.e., if  $t =_\beta t'$  and  $t \not\equiv t'$ , then there is a term  $s$  such that  $t \rightsquigarrow_\beta s$  and  $t' \rightsquigarrow_\beta s$ . One corollary of this is the uniqueness of  $\beta$ -normal forms for terms (provided they have a normal form). Another important corollary for our purposes is that two terms  $t$  and  $t'$  that are in  $\beta$ -nf and syntactically distinct cannot be  $\beta$ -equal, which means that the relation  $=_\beta$  is non-trivial: not all terms are  $\beta$ -equal (Hindley & Seldin, 2008, p. 17).<sup>6</sup>

### 3 What distinguishes ‘good’ from ‘bad’ reductions?

#### 3.1 Problematic reductions

##### 3.1.1 Tonk-reduction

The reduction procedure considered above for  $\rightarrow$  for eliminating maximal formulas, that arise from applying an elimination rule immediately after the corresponding introduction rule, works equally well for our other ‘well-behaved’ connectives (Prawitz, 1965). A comparison with the notorious connective **tonk** might help to see, however, why this is not the case for every connective. **Tonk** was introduced by Prior (1960) as an ad absurdum-attack on the idea of proof-theoretic semantics<sup>7</sup>: If it was only the rules giving the meaning of a connective, no other metaphysically underlying concept, then what would stop the proof-theoretic semanticist from accepting the following rules?

$$\frac{A}{A \text{ tonk } B} \text{ tonkI} \qquad \frac{A \text{ tonk } B}{B} \text{ tonkE}$$

Applying these immediately after each other gives us a derivation from arbitrary  $A$  to arbitrary  $B$ , i.e., our system would trivialize. Additionally, there is no real way to make out a reasonable reduction procedure in this case, which is, of course, due to the fact that

<sup>6</sup>This also implies the consistency of the simply typed  $\lambda$ -calculus, see (Barendregt, 1992, Ch. 2.3) and (Girard, 1989, p. 23).

<sup>7</sup>Although this specific term has been introduced only later in 1991 by Schroeder-Heister (2022), the general idea has been prevalent much longer.

**tonk** violates the inversion principle. This has been one of the ways to give a reason why **tonk** can be considered inadmissible.<sup>8</sup> It should be noted that there are approaches to **tonk**, which do not even consider it inadmissible in principle but which rather question our underlying assumptions about logical consequence on the grounds of which we dismiss **tonk**.<sup>9</sup> Yet, I want to emphasize that even with an argumentation that accepts **tonk**, to my knowledge, there is still no way of giving an acceptable reduction procedure for this connective.

Can this be made explicit with term annotations? Leaving  $\lambda$ -calculus, we can still give term-annotated rules and a corresponding reduction for non-standard connectives, like for a Liar-connective  $L$  for example, as it has been proposed in (Schroeder-Heister, 2012):

$$\frac{t : L \rightarrow \perp}{lt : L} \text{ LI} \qquad \frac{t : L}{l't : L \rightarrow \perp} \text{ LE} \qquad l't \rightsquigarrow_{\perp} t$$

We have  $l$  here serving as a constructor for the introduction rule and  $l'$  as a destructor in the elimination rule. We can do the same for **tonk**, annotating the rules with a constructor  $k$  and a destructor  $k'$ :

$$\frac{t : A}{kt : A \text{ tonk } B} \text{ tonkI} \qquad \frac{t : A \text{ tonk } B}{k't : B} \text{ tonkE}$$

Just like for  $L$  a non-normal term for **tonk** would then be constructed by applying the destructor to the constructor, which is, as for the usual connectives, the result of a derivation containing the conclusion of the introduction rule as the major premise of the elimination rule, i.e.:

$$\frac{\frac{t : A}{kt : A \text{ tonk } B} \text{ tonkI}}{k'kt : B} \text{ tonkE}$$

The usual reduction would be to reduce the term for the conclusion of the elimination rule to the one of the premise of the introduction rule, so analogous to the Liar-reduction:  $k'kt \rightsquigarrow_{\text{tonk}} t$ . However,  $t$  is assigned type  $A$ , while  $k'kt$  is assigned  $B$ . So, if we would accept this reduction, it would mean to accept a reduction relating terms of arbitrarily different types. In the following I want to show that what is wrong with Ekman-reduction is essentially the same as in the case of **tonk**-reduction and on this basis identify what could be a good criterion for reductions of proofs in terms of type theory.

### 3.1.2 Ekman-reduction

Again, Ekman-reduction has the following form:

$$\frac{B \rightarrow A \quad \frac{A \rightarrow B}{B} \rightarrow E \quad \begin{array}{c} \mathcal{D} \\ \vdots \\ A \end{array} \rightarrow E}{A} \rightarrow E \quad \rightsquigarrow \quad \begin{array}{c} \mathcal{D} \\ \vdots \\ A \end{array}$$

The motivation for Ekman to consider this reduction was to give a counterexample to Tennant's (1982) proof-theoretic characterization of paradoxes. According to this, a paradoxical derivation is one that yields a non-normalizable derivation of  $\perp$ . Tennant considers several examples, like versions of the Liar paradox, Curry's paradox or Russell's

<sup>8</sup>Since the **tonk**-rules do not adhere to the inversion principle, they are not in *harmony*. On this notion as a criterion for acceptable connectives, see, e.g., (Tennant, 1978), (Dummett, 1991), (Read, 2010), (Francez & Dyckhoff, 2012), and (Tranchini, 2015).

<sup>9</sup>Cook (2005) and Ripley (2015), e.g., argue like this in claiming that if we do not assume a transitive consequence relation, then an extension with **tonk** would not yield inconsistency. See also (Wansing, 2006), however, where it is shown that the problems of **tonk** avoided in a non-transitive system can be recreated by other **tonk**-like connectives.

paradox, which all have this feature in common and of course, contain some special rules for the respective paradoxical connectives. Ekman (1994; 1998) gives an example of a derivation of  $\perp$ , though, not containing any other rules than the usual ones for implication but which still, if we accept Ekman-reduction that is, could not be brought into normal form because as with Tennant's examples the reduction sequences are looping. Thus, he concludes that Tennant's criterion does not capture a genuinely *paradoxical* feature of the derivations considered, since with Ekman-reduction we could get such a derivation, as well, without containing any paradoxical elements. There have been attempts to show that this can be avoided by using a different representation of the rules, e.g., in (von Plato, 2000) by using general elimination rules in ND showing that such a derivation can be brought to a normal form or in (Tennant, 2021) with rules in sequent calculus showing that we get a cut-free derivation in such a system. This does not stand in opposition to what I am focussing on in this paper, though. Note that these 'solutions' to the so-called *Ekman-paradox* do concede that Ekman-reduction is permissible, since only by using it, we get into this infinite loop of reduction sequences. If we reject Ekman-reduction for independent reasons, for which I will argue in this paper, then Ekman-paradox is no problem either.

The problem with this reduction, which Schroeder-Heister and Tranchini (2017) point out and neatly prove, is the following: if we allow Ekman-reduction (plus always assuming for now that proofs related via reductions can be identified), then we would be forced to identify every derivation of a formula with every other derivation of the same formula, i.e., there would be no basis to distinguish different derivations other than their obvious syntactic difference. We would have to commit to them all representing one and the same proof. It is on these grounds of proof identity that Schroeder-Heister and Tranchini argue that Ekman-reduction should not be counted as an acceptable reduction. I want to show now that annotating Ekman-reduction with terms shows something else that is essentially problematic with this reduction.

In our term-annotated system the derivation to which Ekman-reduction is applied is the following:<sup>10</sup>

$$\frac{y : B \rightarrow A \quad \frac{x : A \rightarrow B \quad \frac{t : A}{\mathcal{D}}}{App(x, t) : B} \rightarrow E}{App(y, App(x, t)) : A} \rightarrow E$$

So the Ekman-reduction procedure for terms would be:

**Ekman-reduction:**  $App(y, App(x, t)) \rightsquigarrow_{Ekman} t$

In the specific case above this reduction seems fine. However, the problem with it, as opposed to the known  $\beta$ -reductions, is that it is too unspecific concerning the term structure. With the same terms (since in Curry-style the types are not part of the syntactical structure of the terms) the following derivation could be constructed:<sup>11</sup>

$$\frac{y : B \rightarrow A \quad \frac{x : (A \rightarrow A) \rightarrow B \quad \frac{t : A \rightarrow A}{\mathcal{D}}}{App(x, t) : B} \rightarrow E}{App(y, App(x, t)) : A} \rightarrow E$$

The derivation is fine but the reduction would be problematic. In this case it is clear that  $A \not\equiv A \rightarrow A$ , i.e., it cannot be the case that the term for  $A$  and the term for  $A \rightarrow A$  constructed out of the same type context in  $\mathcal{D}$  are syntactically the same. Using a concrete

<sup>10</sup>Note that Schroeder-Heister and Tranchini also consider a more general form of this reduction in their paper in that  $A \rightarrow B$  and  $B \rightarrow A$  are not assumptions but are derived formulas. Since this would not change the results here, I will stick to the original form, though.

<sup>11</sup>Still, there are reasons why Curry-style typing is preferable to Church-style, see section 5.

example, we can show why allowing this reduction can create a problem. Consider the following derivation:

$$\frac{y : \tau \rightarrow \rho \quad \frac{x : (\sigma \rightarrow \sigma) \rightarrow \tau \quad \frac{[z : \sigma]}{\lambda z.z : \sigma \rightarrow \sigma} \rightarrow I}{\lambda z.z : \sigma \rightarrow \sigma} \rightarrow E}{App(x, \lambda z.z) : \tau} \rightarrow E}{App(y, App(x, \lambda z.z)) : \rho} \rightarrow E$$

So,  $App(y, App(x, \lambda z.z))$  would Ekman-reduce to  $\lambda z.z$ . However, no type assigned to  $\lambda z.z$  can be atomic, i.e.,  $\lambda z.z : \rho$  is impossible. The problems arising for these reductions relate to questions of so-called *type preservation*, *typechecking*, and *type reconstruction*, which I will discuss in the next section.

### 3.2 Subject reduction and type reconstruction

The problem of **tonk**- and Ekman-reduction seems to be that, unlike the  $\beta$ -reductions, they are not type preserving. Let us briefly take a look at this property and its significance for reduction procedures. Sometimes the expressions *subject reduction* and *type preservation* are used synonymously. However, type preservation describes a broader concept than subject reduction, since the latter only says that types are preserved when terms (i.e., “subjects”) are *reduced*, whereas type preservation can also be used to describe a property of subject *expansions*. So, we will distinguish this terminology here.<sup>12</sup> The subject reduction theorem for the proof system with  $\lambda$ -terms we consider states the following (Sørensen & Urzyczyn, 2006, p. 59):

**Subject Reduction Theorem:** If  $\Gamma \vdash t : A$  and  $t \rightsquigarrow_{\beta} t'$ , then  $\Gamma \vdash t' : A$ .

Subject expansion, on the other hand, does not hold for this system in general, i.e., it is not the case that if  $t \rightsquigarrow_{\beta} t'$  and  $t' : A$ , then  $t : A$ , meaning that the set of types assigned to a term is not invariant under conversion in general (see, e.g., (Barendregt, 1992, p. 41); (Hindley & Seldin, 2008, p. 170) for counterexamples).

The examples given in the previous section clearly show that subject reduction does not hold for **tonk**- and Ekman-reduction, i.e., it is not the case that whenever  $t : A$  and  $t \rightsquigarrow_{Ekman/\mathbf{tonk}} t'$ , then  $t' : A$ . We can also say that the contractum does not *typecheck* at every type the redex typechecks at. Typechecking is something that needs to be considered in Curry-style type systems (see, e.g., Sørensen & Urzyczyn, 2006, p. 60) and is about deciding whether or not  $\Gamma \vdash t : A$  holds, for a given context  $\Gamma$ , a term  $t$  and a type  $A$ . We can express typechecking in the following form then:

**Typechecking:**  $t$  *typechecks* at  $A$  iff  $\Gamma \vdash t : A$  holds, for a given context  $\Gamma$ , a term  $t$  and a type  $A$ .

As can be seen above, there are cases with Ekman-reduction (and for **tonk** it is even more obvious) in which it is *impossible* to assign  $t'$  the type assigned to  $t$ . If we understand types like labels telling us the combinations that can safely be made with a term, then we can understand subject reduction as saying that a term will not become ‘less safe’ during a reduction, i.e., when performing a computation on a term, this term cannot turn from a well-typed into an ill-typed one (Hindley & Seldin, 2008, p. 168). Subject reduction thus establishes the correctness of our system of type assignment (Sørensen & Urzyczyn, 2006, p. 59). It seems, therefore, that maintaining subject reduction would certainly be a desirable feature for reduction procedures.

So, is subject reduction a good criterion to measure the acceptability of reduction procedures? To deal with this question we need to determine whether non-type-preserving

<sup>12</sup>If in the following a reduction is stated (not) to be type preserving, this means that it enjoys (no) subject reduction, i.e., reductions are to be understood in the one-directed sense without looking at the other direction of expansions.

reductions necessarily lead to trivialization of the system. In other words, are there systems which can contain reduction procedures that are not type preserving but yet do not trivialize the reducibility relation? Although it looks like a promising criterion for reductions, it actually seems to be the case that failure of subject reduction need not necessarily cause trivialization. To wit, it does not seem impossible that there could be a type theory with a reduction that is not type preserving without relating terms of arbitrary types but, e.g., only of types which are equivalent (i.e., interderivable formulas).<sup>13</sup>

The actual problem with the `tonk`-/Ekman-reductions, though, leading to trivialization of the system, can be identified when looking at type reconstruction for their redexes. Type reconstruction is used to decide the typability of terms (Sørensen & Urzyczyn, 2006, p. 60):

**Type Reconstruction:** Given term  $t$ , decide if there is a context  $\Gamma$  and a type  $A$ , such that  $\Gamma \vdash t : A$ .

This can be achieved using a type reconstruction algorithm, which is simply based on the type assignment rules that are used. Since these are just given by the annotated inference rules of our system, they are syntax-oriented. This means that we should be able to figure out the *principal types* of terms, i.e., figure out the derivation by reconstructing bottom-up the term using the type assignment rules. To give an example of a successful type reconstruction, let us consider the one for the redex  $App(\lambda x.t, s)$  resulting from our  $\rightarrow$ -rules starting with assigning it an arbitrary type  $B$ . We write ‘?’ whenever this part of the type is syntactically undetermined in this step of the reconstruction. Two occurrences of ‘?’ in the same step mean that, although their structure is undetermined, they must be filled in by the same type symbol. In the next step we are to use a ‘fresh’ type symbol for ‘?’.

**Type reconstruction for  $App(\lambda x.t, s)$ :**

$$\frac{\lambda x.t : ? \rightarrow B \quad \begin{array}{c} \mathcal{D}' \\ \vdots \\ s : ? \end{array}}{App(\lambda x.t, s) : B} \rightarrow E \qquad \frac{\begin{array}{c} [x : A] \\ \vdots \\ t : B \end{array} \quad \begin{array}{c} \mathcal{D}' \\ \vdots \\ s : A \end{array}}{\lambda x.t : A \rightarrow B \quad s : A} \rightarrow I \quad \frac{}{App(\lambda x.t, s) : B} \rightarrow E$$

As we can see, the type reconstruction proceeds in such a way that *we have to* assign contractum  $t$  the same type as the redex. The structure of the redex and the connected type assignment rules lead to an exactly determined type reconstruction, which cannot ‘go wrong’ concerning the relation between types of redex and contractum.

### 3.3 Criterion for acceptable reductions

The problem with allowing a reduction such as the one for `tonk`, however, can be shown by a type reconstruction of the non-normal term  $k'kt$ , assuming  $k'kt \rightsquigarrow_{\text{tonk}} t$  as a reduction, as motivated above. If we assign  $k'kt$  an arbitrary type  $B$ , then the only information this gives us for  $kt$  is that its type must be of the form “? `tonk`  $B$ ”. Consequently,  $t$  can be assigned an arbitrary type. This means that the types of redex and contractum are arbitrarily independent of each other, which is exactly the core of the problem with a reduction for `tonk`.

<sup>13</sup>An anonymous reviewer pressed the point here that it seemed unnecessary to look for a weaker criterion than subject reduction since with it we do get rid of the problematic cases. However, this would only amount to give a sufficient criterion, not a necessary one. Indeed, if a reduction enjoys the property of subject reduction, then it will be deemed acceptable and the criterion I will propose here will not stand against that. But that does not exclude the possibility that there might be reductions that are acceptable without having this property (see section 3.4).



**Type reconstruction for  $k'kt$ :**

$$\frac{kt : ? \text{ tonk } B}{k'kt : B} \text{ tonkE} \qquad \frac{t : A}{kt : A \text{ tonk } B} \text{ tonkI} \text{ tonkE} \qquad \frac{}{k'kt : B} \text{ tonkE}$$

With type reconstruction it also becomes evident that the same problem as with **tonk** prevails with Ekman-reduction. We are doing a type reconstruction for the redex again. If we assign  $App(y, App(x, t))$  an arbitrary type  $A$ , then we can reconstruct bottom-up the following derivation in which a new type variable is used whenever it is independent from the ones already used (skipping the step-by-step illustration with ‘?’):

**Type reconstruction for  $App(y, App(x, t))$ :**

$$\frac{y : B \rightarrow A \quad \frac{x : C \rightarrow B \quad \frac{t : C}{\mathcal{D}}}{App(x, t) : B} \rightarrow E}{App(y, App(x, t)) : A} \rightarrow E$$

Again, such a reduction allows reducing a term of one type to one of an *arbitrary* other; one that is arbitrarily unrelated in the type reconstruction from the type of the term that is reduced.

This arbitrariness cannot arise with the standard  $\beta$ -reductions and, importantly, there are also other non-standard reductions which are well-behaved with respect to this feature, i.e., this is not simply to say that  $\beta$ -reductions are the only acceptable reductions. For instance, if we compare **tonk**-reduction to the Liar-reduction given above, of course, they look very similar. But in the Liar case type reconstruction quickly shows that this reduction is well-behaved, while the **tonk**-reduction is not.

So, what we are actually asking for is what I will call a ‘weak’ subject reduction:

**Weak Subject Reduction:**

- (i) If  $\Gamma \vdash t : A$  and  $t \rightsquigarrow t'$ , then  $\Gamma \vdash t' : A$ , or
- (ii) if  $\Gamma \vdash t : A$ ,  $t \rightsquigarrow t'$  and  $\Gamma \not\vdash t' : A$ , then it is not the case that  $\Gamma \vdash t' : B$  for arbitrary  $B$ .  $B$  is considered *arbitrary* iff the rules of type assignment do not determine the type reconstruction of  $t$  in a way that  $B$  is related to  $A$ .

This is what I propose to demand as a criterion for a reduction to be acceptable: it should enjoy the property of weak subject reduction. To reformulate it in other words, what we demand is that for the case that ‘full’ subject reduction, i.e., clause (i), fails,  $\Gamma \vdash t' : B$  holds only for those  $B$ , which the rules of type assignment relate to  $A$  in the type reconstruction of  $t$ . This criterion ensures that whenever subject reduction holds, weak subject reduction holds as well, i.e., failure of weak subject reduction also implies failure of ‘full’ subject reduction. That is important because it means that not meeting this desideratum only rules out the ‘bad’ reductions. The ones for which ‘full’ subject reduction holds, which, as we said above, is deemed to be sufficient for ‘acceptable’ reductions, cannot be ruled out by that criterion. Also, note that failure of weak subject reduction does not necessarily mean that there is something wrong with the rules of type assignment in question. In the case of Ekman-reduction there is nothing wrong with the rules, since the only rules used are the ones for  $\rightarrow$  and those are fine for the  $\beta$ -reduction. It rather shows that the reduction generated on grounds of these rules is misbehaved: it may work for specific types but it cannot be generalized in the same way ‘proper’ reductions can.

Our way of *checking* whether weak subject reduction holds or not is then via type reconstructions in the way described above: We conduct a type reconstruction for a term that would count as non-normal under this reduction, i.e., a redex, choosing ‘fresh’ types

whenever the type assignment rules allow this. If the resulting types of redex and contractum occurring in this reconstruction are of arbitrarily different, unrelated types, then weak subject reduction fails and this means that this reduction should be rejected. Thus, we do not only have a clear criterion of what distinguishes acceptable from unacceptable reductions but also a fairly simple way of testing this by the respective type reconstruction. That can be considered an advantage when comparing it to Schroeder-Heister and Tranchini’s way of showing how Ekman-reduction leads to unacceptable consequences, which they do by giving a very well-thought-out example of certain derivations leading to these consequences. This is a very clever and sophisticated way, for sure, but one has to be able to come up with these examples in the first place. Here, on the other hand, we have a systematic procedure of checking whether a reduction is acceptable or not.

### 3.4 Type theory of core logic - another problematic case?

In the following I want to give a concrete example of a reduction which is not type preserving, i.e., does not enjoy ‘full’ subject reduction, but still does not necessarily have to be dismissed as a ‘bad’ reduction.<sup>14</sup> The reduction is presented by Ripley (2020) as part of an interesting typed term calculus for Tennant’s Core Logic, i.e., an intuitionistic relevant logic. The calculus, called *Core Type Theory*, is interesting because it displays some very unusual features, while at the same time it is - at least in some respects - quite well-behaved.

According to Ripley the system maintains a similar correspondence to the implication-negation-fragment of core logic as the one established by the Curry-Howard correspondence between the simply typed  $\lambda$ -calculus and intuitionistic logic. In the proof system that he presents, next to formulas and connectives we have  $\odot$ , which is related to negation but should not be considered as something like  $\perp$ .  $\odot$  is neither a formula nor a connective, i.e., it cannot be used to form any complex formulas, but rather, it is understood as a “structural marker that interacts with the connective rules” in a way specified by the proof system (Ripley, 2020, p. 112). One of the things to note is that in core type theory in addition to the usual case where terms are of certain types (served by the formulas), here terms can also have  $\odot$  instead of a type, in which case Ripley speaks of *exceptional terms*. Also, Ripley uses Church-style typing, i.e., the types and  $\odot$  are part of the syntax of the terms.

Ignoring differences in notation, redex and contractum are defined in the same way as above, i.e.,  $App(\lambda x.t, s)$  as redex and the corresponding term  $t[s/x]$  as contractum.<sup>15</sup> Relevant for our purpose is that the reduction procedure fails to be type preserving because it can happen that a typed term, on which we perform a reduction procedure, has an exceptional term as contractum. Also, the system is not confluent, which means that normal forms are not unique, i.e., two syntactically distinct terms in normal form do not necessarily belong to two distinct equivalence classes generated by this reduction. The system is indeed trivializing in the sense that *if* we assume proof identity via the equivalence relation induced by its reduction procedure, then every term would have to be identified with every other term (Ripley, 2020, p. 128). What must be stressed, however, is that there is only one non-type-preserving direction that is possible, namely from typed

<sup>14</sup>Another example could be found in (Wansing, 1993), where a type theory for Nelson’s logic with strong negation,  $\mathbb{N}4$ , is given, which identifies terms of type  $A$  with terms of type  $\sim\sim A$ . A concrete reduction is not formulated there but it is likely that it would have similar features as the one discussed in this section, since in such a system there would have to be rules of type assignment for  $\sim$  which in some way relate  $A$  and  $\sim\sim A$ .

<sup>15</sup>It may be noted here that in the definition of those, the differences between Curry- and Church-style typing are blurred somehow because the terms used in the definition are not typed, which would be the usual thing to be done in Church-style, see (Hindley, 1997, p. 26) and (Troelstra & Schwichtenberg, 2000, p. 13). Sørensen and Urzyczyn (2006), who Ripley refers to in this context, leave out the types in their definition, as well, however, they say themselves that their ‘Church-style’ is actually “halfway between the Curry style and the ‘orthodox’ Church style” (Sørensen & Urzyczyn, 2006, p. 66).

to exceptional terms. We cannot go from terms of one type to terms of another type or from exceptional terms to typed terms. This is one of the preservation properties this system still has. Another is that the reduction can never lead to new free variables, i.e., the set of free variables in a redex is a (possibly proper) superset of the free variables in its contractum (Ripley, 2020, p. 116). This is the same as in the simply typed  $\lambda$ -calculus.

Since this is no system of type assignment but a typed system à la Church, the issue of type reconstruction can actually not be raised (Sørensen & Urzyczyn, 2006, p. 66). However, it seems rather unproblematic to convert Ripley’s system into a system in which types and  $\odot$  are assigned to terms according to the inference rules that are given. The rules and reduction for  $\rightarrow$  (he additionally considers rules for negation) would then look like this:

$$\frac{[x : A] \quad \vdots \quad t : B}{\lambda x.t : A \rightarrow B} \rightarrow I \quad \frac{[x : A] \quad \vdots \quad t : \odot}{\lambda x.t : A \rightarrow B} \rightarrow I! \quad \frac{\Gamma \quad \Delta \quad [y : B] \quad \vdots \quad s : A \rightarrow B \quad \vdots \quad t : A \quad \vdots \quad r : C}{r[App(s, t)/y] : C} \rightarrow E$$

$$App(\lambda x.t, s) \rightsquigarrow_{\beta} t[s/x]$$

Since the inference rules are not as determined as our standard rules,<sup>16</sup> it is clear that type reconstruction<sup>17</sup> cannot be conducted in such a way that it yields a determinate result as with the standard rules. Importantly, however, neither does it result in complete arbitrariness of the kind we have seen with Ekman-reduction or **tonk**. What can happen indeed, is that due to the two  $\rightarrow$ -I rules, we have *two possible* paths in the type reconstruction, but that’s it:<sup>18</sup>

$$\frac{\lambda x.t : ? \rightarrow B \quad \mathcal{D}' \quad \vdots \quad s : ?}{App(\lambda x.t, s) : B} \rightarrow E \quad \frac{[x : A] \quad \vdots \quad \mathcal{D} \quad t : B/\odot \quad \mathcal{D}' \quad \vdots \quad s : A}{\lambda x.t : A \rightarrow B \quad \rightarrow I/\rightarrow I! \quad App(\lambda x.t, s) : B} \rightarrow E$$

The two paths are marked by the step in red. Everything else will be exactly the same, though. Since the reduction is  $App(\lambda x.t, s) \rightsquigarrow_{\beta} t[s/x]$ , it can happen that the redex reduces to a contractum which does not have the same type (neither does it have *another* type, though, because  $\odot$  is no type at all). So, this means that the reduction in this system is not type preserving, i.e., subject reduction fails. However, *weak* subject reduction holds since it cannot reduce to an arbitrary type. The contractum will be either of the same type as the redex or it will be assigned  $\odot$ , which *is* related to  $B$  by the type assignment rules: thus, to this extent the type reconstruction is determined.

Therefore, we have a reduction in this system which is at least partially well-behaved. On the one hand, confluence and subject reduction fail and if we would like the equivalence relation induced by reductions to give us proof identity, the reduction in this type theory would certainly not be suitable, since it trivializes identity of terms. On the other hand, type reconstruction can be conducted in an ordered manner without the possibility of yielding arbitrary results. Thus, the cases in which subject reduction fails are not completely arbitrary concerning the types, since it is not possible, as opposed to Ekman- and **tonk**-reduction, that a well-typed term reduces to a term of an arbitrarily different type. An anonymous reviewer raised doubts about the acceptability of this system because the identity of terms would be trivialized by the reductions, demanding that disallowing this

<sup>16</sup>As can be seen, we have two  $\rightarrow$ -introduction rules.

<sup>17</sup>If we want to be very precise, we would have to speak of “*hat* reconstruction”, “rules of *hat* assignment”, etc. since this is Ripley’s terminology for including both types and  $\odot$ . For simplicity, though, and because the criterion of weak subject reduction would still be met under such a reformulation, we will stick to the usual terminology.

<sup>18</sup>Note that although in core logic we have a generalized form of the elimination rule for  $\rightarrow$ , the instance of this rule here is the usual Modus Ponens since Ripley defines a redex being of the form  $App(\lambda x.t, s)$ .

should rather be our minimal criterion for the acceptability of a system. Note here that we must distinguish between the equivalence relation induced by the reductions and the reduction relation itself. While the former is certainly too permissive to be interesting for a philosophical interpretation of the proof theory, the latter can still be recognized to be at least so well-behaved that it does not lead to an Ekman-tonk-ish kind of trivialization, which is the kind we are worried about for reasons to be discussed in the following section.

## 4 Philosophical implications: Reduction procedures and meaning of proofs

One of Prawitz's most important conjectures in this context is that, since the reductions induce an equivalence relation and two derivations should be considered to represent the same proof iff they are equivalent, proofs relating via these reductions are identical in nature.<sup>19</sup> This means in general that one and the same proof may be linguistically represented by different derivations and that in natural deduction a derivation in *normal form* is the most direct form of representation of its denotation, i.e., the represented proof object.<sup>20</sup>

Failure of weak subject reduction means to have reductions that relate terms of arbitrarily different types, i.e., proofs of arbitrarily different formulas. If we consider reductions to induce identity of proofs, a feature that ultimately results in having to identify proofs of arbitrarily different formulas would certainly be undesirable. However, there is no necessity to subscribe to this identity theory of proofs. There are other views on theories about identity of proofs on the market, of course, or it is also possible to argue like Tennant (2021, p. S599), who seems to be a bit of an agnostic when it comes to this question. He indicates, though, in response to the proposal made in (Schroeder-Heister & Tranchini, 2017) to discard Ekman-reduction because it leads to a trivialization of proof identity, that we do not know enough about identity of proofs to use it as a criterion for other conceptions.<sup>21</sup> However, reductions can also be conceived of as calculations, evaluations, or interpretations of the given program, as discussed in section 2. I will argue here that if we go for the latter conception of reductions, failure of weak subject reduction still remains a problem, even if it is not problematic for the identity of proofs anymore. While this would be a problem for the *denotation* of proofs, I want to show that the arbitrariness is a problematic feature also concerning the *sense* of proofs.

Tranchini (2016) argues that only proofs which contain connectives for which reduction procedures are available can have sense. He bases his argumentation on the Prawitzian tradition that derivations in normal form can be identified with the proof objects, i.e., their denotation, and the fact that the reductions are the instruments with which we can bring a derivation to its normal form. If reductions for terms are considered to be decisive for the meaning of proofs, it seems that we should be clear about the question of the

<sup>19</sup>What is left undecided in Prawitz's remarks (1971, p. 257) is whether the  $\beta$ -reductions are the only conversions preserving identity of proofs or whether expansion operations (corresponding to  $\eta$ -expansions) and permutative conversions for  $\vee$ -elimination and  $\exists$ -elimination should be considered as well. He seems to lean towards accepting at least the expansions, while Martin-Löf (1975, pp. 100f.) discards both kinds of operations for identity preservation. Girard (1989, pp. 16, 73), on the other hand, includes  $\eta$ -expansions but is highly sceptical w.r.t. the permutative conversions when it comes to the question of identifying the 'real objects' represented by the ND derivations. Since I am concerned only with *reductions* here and not conversions in general, I will leave this issue as it is.

<sup>20</sup>This Fregean formulation can be found in (Tranchini, 2016), where this is meant to explicate Prawitz's and Dummett's conceptions on these matters.

<sup>21</sup>Examples for other approaches to proof identity would be Straßburger's (e.g., 2019) based on graphical proof practice, like proof nets for linear logic or what he calls a 'combinatorial' approach for classical logic. Another one is Wansing's (in press) approach, where a notion of identity between derivations is defined on the basis of taking both the notion of proofs as well as *disproofs* as primitive. On this account proofs of certain formulas can be identified with disproofs of other, in specific ways related formulas and based on this a bilateralist notion of synonymy between formulas is defined subsequently.

present paper: What are the conditions of acceptable reduction procedures? In (Ayhan, 2021) the general assumption from Tranchini is retained that the connectives appearing in a derivation need to have acceptable reductions in order for the derivation to have sense at all and based on this an approach with  $\lambda$ -term-annotated proof systems is motivated to spell out what the sense of derivations consists in. It is argued that in a term-annotated setting the denotation of derivations is represented by the end-term<sup>22</sup> of the derivation in normal form, since this term encodes the ultimate proof. The *sense* of a derivation, on the other hand, consists in the *set of terms occurring within the derivation* because those terms encode the intermediate steps in the construction of the complex end-term encoding the conclusion (Ayhan, 2021, p. 578). Thus, these terms reflect the operations used in the derivation, i.e., they reflect the way that is taken to get to the denotation. Since they determine how the end-term is built up, they can be seen as encoding a procedure, which, finally, yields the end-term. This seems in accordance with what, e.g., Dummett (1973, pp. 232, 323, 636) (a “procedure” to determine the denotation), Girard (1989, p. 2) (“a sequence of *instructions*”) or Horty (2007, pp. 66-69) (“senses as procedures”) say about Fregean sense (Girard even in the context of relating this to the “proofs as programs” conception).<sup>23</sup>

According to Frege, what is crucial, is that the signs (here: the syntax) uniquely determine the sense and the sense uniquely determines the denotation. What can happen, though, is of course the classic example of Hesperus and Phosphorus, where there is the same underlying denotation but different senses attached to different syntax (i.e., different words). In the context of proofs this would be indicated by different terms used within the derivations, ending, however, on the same end-term or being reducible to the same end-term. The following example illustrates this with a derivation in non-normal form reducing to the other, which is in normal form, since  $App(\lambda y.\lambda x.x, \lambda y.y) \rightsquigarrow_{\beta} \lambda x.x$ :

$$\frac{\frac{\frac{[x : p]}{\lambda x.x : p \rightarrow p} \rightarrow I}{\lambda y.\lambda x.x : (q \rightarrow q) \rightarrow (p \rightarrow p)} \rightarrow I}{App(\lambda y.\lambda x.x, \lambda y.y) : p \rightarrow p} \rightarrow I \quad \frac{\frac{[y : q]}{\lambda y.y : q \rightarrow q} \rightarrow I}{\lambda x.x : p \rightarrow p} \rightarrow E \quad \frac{[x : p]}{\lambda x.x : p \rightarrow p} \rightarrow I$$

The relation of these conceptions and the (un)acceptability of reduction procedures is the following now. Whether or not we see the reductions as generating identity, or ‘merely’ in this directed way as calculations, makes a difference concerning the denotation but *not* concerning the sense. We could use the theory described here but only equate terms over  $\alpha$ -conversion, for example. The derivations above, one reducing to the other, would not be identified anymore in this case but the senses would remain unchanged. They would not be identified because the denotation is referred to by the end-terms and if we do not assume identity over  $\beta$ -reductions, then these terms could not be identified, i.e., they would point to different proof objects. The senses, though, consist in both cases (whether or not we assume  $\beta$ -equality for the end-terms) of the terms occurring within the derivations, i.e., they are different from each other in both cases but each for itself does not change by that assumption about the denotation of the proofs. It would still hold that the sense determines the reference, in that there cannot be one sense leading to different denotations, and, importantly, that the syntax determines the sense. This can only be claimed to hold, however, if the rules (the syntax) determine the type reconstruction for the redexes of reductions (the sense) to the extent that types of redex and contractum are not arbitrarily unrelated. Otherwise, it cannot be said that the syntax determines the sense. That needs to be the case, though, since meaning must be governed by rules. It cannot be arbitrarily generated. So, even if we do not accept the assumption that reductions

<sup>22</sup>The term decorating the formula that is proven.

<sup>23</sup>Of course, there are other approaches on the Fregean sense like Evans’ (1982), on whose conception the interpretation given here could not be considered since lacking denotation would mean lacking sense as well. However, I do not find that interpretation very convincing, especially not in this context but also not in general (see also (Fitch & Nelson, 2018) on the problems of this conception).

generate equivalence relations over which proofs can be identified, it still makes sense to disallow reductions which render the derivations they are related to meaningless.

## 5 Remarks on possible objections and Church- vs. Curry-style typing

An objection that may be raised against the present approach in general is that it does not actually tell us anything interesting or important about reductions but rather, that it shows a weakness of the underlying assumption that using such term-annotated proof systems is a good way to go. The argument could be delivered along the following lines: The version of giving Ekman-reduction in a term-annotated form, as presented in this paper, is too generalized to capture what the reduction in original form meant to express. In the original form the assumptions  $A \rightarrow B$  and  $B \rightarrow A$  are clearly essential for having such a reduction but these disappear completely in the reduction for terms considered here. Thus, it is inadmissible to take this reduction, lacking essential features, as a correspondence for the original Ekman-reduction and dismiss it on this basis.

Further, it could be argued that, in order to capture the original Ekman-reduction appropriately, a restriction on the types should be implemented. Such a restriction could be:  $App(y, App(x, t)) \rightsquigarrow_{Ekman} t$  iff  $t$  typechecks at every type  $App(y, App(x, t))$  typechecks at.<sup>24</sup> What you get thereby is basically subject reduction *by definition*. It still might seem a bit more generalized than the original Ekman-reduction because we do not demand that the types are the same but the only way they could differ now is in the variables used for the atomic formulas, i.e., the principal types are always the same. Of course, with such a restriction we would not have the undesirable arbitrariness in type reconstruction and thus, such a reduction would be well-behaved.

However, such a restriction must be rejected as a ‘saviour’ for Ekman-reduction, since implementing it would entirely beg the question of what we wanted to devise here. Firstly, in exactly the same way  $\mathbf{tonk}$ -reduction could be restored: By demanding that  $k'kt \rightsquigarrow_{\mathbf{tonk}} t$  iff  $t$  typechecks at every type  $k'kt$  typechecks at. I do not see, though, that this is what we are philosophically interested in when we want to investigate the nature of reductions. It could not be said anymore, as Prawitz did, that the reductions make the inversion principle explicit if what we are doing is to restrict them by definition to cases in which the inversion principle is maintained. What we are interested in, concerning the question “What are acceptable reductions?”, is not to decide case by case whether it makes a difference to eliminate a certain detour or not, but to have some generalized form about which we can make such a judgment.

What must be considered, secondly, when asking why such a restriction should be rejected in our approach, is that by using it we would abandon basics of Curry-style typing and de facto do Church-style typing instead.<sup>25</sup> One could claim now, thereby raising another objection, that indeed, it simply would be better to use Church-style typing. The related objection would be to say that these features of type reconstruction just show that Curry-style typing has a severe disadvantage over Church-style, namely a looseness of the connection between terms and types, which makes it less beneficial for an approach like the present. In other words, what all of this shows, is not that there is something wrong with certain reductions, but that our typing system is not helpful for

<sup>24</sup>Note that it does not suffice here to demand that there is a type  $A$  such that both  $App(y, App(x, t))$  and  $t$  typecheck at  $A$ . To see why this is not enough, consider the example at the end of section 3.1. There is a type such that  $App(y, App(x, \lambda z.z))$  and  $\lambda z.z$  both typecheck at, namely  $\sigma \rightarrow \sigma$  (if we had  $\sigma \rightarrow \sigma$  instead of  $\rho$  in the example, this would clearly work out).

<sup>25</sup>Whereas in Curry-style the syntax of the terms is independent of types, in Church-style types are part of the syntax of terms. This means that each variable is *uniquely* typed and therefore, e.g.,  $\lambda x^A.x^A$  is a term of type  $A \rightarrow A$  but not of, let's say,  $B \rightarrow B$ . In Curry-style, on the other hand,  $\lambda x.x$  is a term of type  $A \rightarrow A$  for every  $A$ .

this question and that we should rather use a type system à la Church (against which the first objection could not be raised anymore, either).

However, if we are interested in proofs from a philosophical, rather than merely technical, point of view, then Curry-style typing is preferable to Church-style. In Church-style you *will* get invariance of types under conversion but just because of the definition of the language, not because it is an interesting property. All terms are typed, i.e., so are the  $\beta$ -redexes. Thus,  $\beta$ -reduction is restricted w.r.t. types and type changes are prevented (Hindley, 1997, p. 26). It is exactly these features of Church-style language then, which prevent us from asking philosophically interesting questions. Because the types are part of the syntax of the terms and the typing rules are just part of the definition of the language, they cannot be used to answer questions about a more primitive, underlying language like “Is this term typeable/meaningful?”, “Can this term be assigned this or that type?”,... If you think of these questions as applied to proofs, these are philosophically interesting questions, though. With Curry-style we have a language in which those can be asked, in Church-style they are prevented simply by the definition of the language.

## 6 Conclusion

What cannot be provided by our analysis here is an exhaustive list of properties that reductions need to have in order to count as ‘good’ because this seems to depend on the role one wants a reduction to fulfill, which differs in the literature. All we can do is to draw a distinct line of what *unacceptable* reductions are: reductions which do not enjoy the property of *weak subject reduction*, that is, which yield the possibility of type reconstructions in which redex and contractum are arbitrarily independent of one another. Further, it can be claimed that if ‘full’ subject reduction fails, this does not necessarily need to lead to the exclusion of such a reduction. It is a reason to be careful about identifying terms via the reduction, though. Within the framework I outlined in this paper we have three kinds of reductions: firstly, the ones that are clearly well-behaved, like  $\beta$ -reductions. They have what seem to be very desirable features, like having the Church-Rosser-property, preserving types, etc. Secondly, we have reductions which are clearly not well-behaved. Those would be Ekman-reduction, or **tonk**-reduction, or any reduction which does not allow for a meaningful system because it arbitrarily connects terms of different types. Thirdly, we have reductions in between those two categories. These would be reductions like the example we saw in section 3.4; ones which may lack desirable features but are still well-behaved enough that they need not necessarily be excluded. Whether or not one wants to accept them, then depends on the underlying philosophical theory (e.g., about identity of proofs) one is subscribing to.

What I showed in this paper is that the question of what makes up acceptable reductions is neither trivial nor easy to answer in a positive way. Thus, I make do with a negative answer, just like Schroeder-Heister and Tranchini do in their elaborate analysis of the topic in saying that acceptable reductions are not to yield an equivalence relation that trivializes the identity of proofs. While I agree with their analysis, I aimed at going a step further and show that even if one does not agree with the underlying assumption that reductions induce an identity relation for proofs, there are certain reductions, like Ekman-reduction, which still have to be considered problematic. The main point is that having to identify all proofs of the same formula is surely undesirable but it is all about the denotation. However, if we have to commit to a notion of reductions according to which terms of a certain type reduce to terms of arbitrarily unrelated types, then such a system cannot be considered rule-generated anymore, and thus, not meaningful.

## 7 References

Ayhan, S. (2021). What is the meaning of proofs? A Fregean distinction in

- proof-theoretic semantics. *Journal of Philosophical Logic*, 50, 571-591. doi: <https://doi.org/10.1007/s10992-020-09577-2>
- Baader, F., & Nipkow, T. (1998). *Term rewriting and all that*. Cambridge: Cambridge University Press.
- Barendregt, H. (1992). Lambda calculi with types. In S. Abramsky, D. M. Gabbay, & T. S. E. Maibaum (Eds.), *Handbook of logic in computer science* (Vol. 2, pp. 117–309). Oxford: Oxford University Press.
- Cook, R. T. (2005). What’s wrong with tonk(?). *Journal of Philosophical Logic*, 34, 217–226. doi: <https://doi.org/10.1007/s10992-004-7805-x>
- Dummett, M. (1973). *Frege: Philosophy of language*. New York: Harper & Row.
- Dummett, M. (1991). *The logical basis of metaphysics*. London: Duckworth.
- Ekman, J. (1994). *Normal proofs in set theory* (Ph.D. dissertation). University of Göteborg.
- Ekman, J. (1998). Propositions in propositional logic provable only by indirect proofs. *Mathematical Logic Quarterly*, 44, 69–91. doi: <https://doi.org/10.1002/malq.19980440105>
- Evans, G. (1982). *The varieties of reference*. Oxford: Clarendon Press.
- Fitch, G., & Nelson, M. (2018). Singular propositions. In E. N. Zalta & U. Nodelman (Eds.), *The Stanford Encyclopedia of Philosophy* (Spring 2018 ed.). Metaphysics Research Lab, Stanford University. doi: <https://plato.stanford.edu/archives/spr2018/entries/propositions-singular/>
- Francez, N., & Dyckhoff, R. (2012). A note on harmony. *Journal of Philosophical Logic*, 41, 613–628. doi: <https://doi.org/10.1007/s10992-011-9208-0>
- Girard, J.-Y. (1989). *Proofs and types*. Cambridge: Cambridge University Press.
- Hindley, J. R. (1997). *Basic simple type theory*. Cambridge: Cambridge University Press.
- Hindley, J. R., & Seldin, J. P. (2008). *Lambda-Calculus and Combinators: an Introduction*. Cambridge: Cambridge University Press.
- Horty, J. (2007). *Frege on definitions: A case study of semantic content*. Oxford: Oxford University Press.
- Martin-Löf, P. (1975). About models for intuitionistic type theories and the notion of definitional equality. In S. Kanger (Ed.), *Proceedings of the Third Scandinavian Logic Symposium* (pp. 81–109). Amsterdam: North Holland.
- Prawitz, D. (1965). *Natural deduction: A proof-theoretical study*. Stockholm: Almqvist & Wiksell.
- Prawitz, D. (1971). Ideas and results in proof theory. In J. E. Fenstad (Ed.), *Proceedings of the Second Scandinavian Logic Symposium* (pp. 235–307). Amsterdam: North Holland.
- Prior, A. N. (1960). The runabout inference-ticket. *Analysis*, 21(2), 38–39.
- Read, S. (2010). General-elimination harmony and the meaning of the logical constants. *Journal of Philosophical Logic*, 39, 557–576. doi: <https://doi.org/10.1007/s10992-010-9133-7>
- Ripley, D. (2015). Anything goes. *Topoi*, 34, 25–36. doi: <https://doi.org/10.1007/s11245-014-9261-8>
- Ripley, D. (2020). Strong normalization in core type theory. In I. Sedlár & M. Blicha (Eds.), *The Logica Yearbook 2019* (pp. 111–130). London: College Publications.
- Schroeder-Heister, P. (2012). Proof-theoretic semantics, self-contradiction, and the format of deductive reasoning. *Topoi*, 31, 77–85.
- Schroeder-Heister, P. (2022). Proof-theoretic semantics. In E. N. Zalta & U. Nodelman (Eds.), *The Stanford Encyclopedia of Philosophy* (Winter 2022 ed.). Metaphysics Research Lab, Stanford University. doi: <https://plato.stanford.edu/archives/spr2018/entries/proof-theoretic-semantics/>
- Schroeder-Heister, P., & Tranchini, L. (2017). Ekman’s paradox. *Notre Dame Journal of Formal Logic*, 58(4), 567–581.



- Schroeder-Heister, P., & Tranchini, L. (2018). How to Ekman a Crabbé-Tennant. *Synthese*. doi: <https://doi.org/10.1007/s11229-018-02018-3>
- Sørensen, M., & Urzyczyn, P. (2006). *Lectures on the Curry-Howard Isomorphism*. Amsterdam: Elsevier Science.
- Straßburger, L. (2019). The problem of proof identity, and why computer scientists should care about Hilbert's 24th problem. *Philosophical Transactions of the Royal Society A*, 377(2140). doi: <https://doi.org/10.1098/rsta.2018.0038>
- Tennant, N. (1978). *Natural logic*. Edinburgh: Edinburgh University Press.
- Tennant, N. (1982). Proof and paradox. *Dialectica*, 36, 265–296.
- Tennant, N. (1995). On paradox without self-reference. *Analysis*, 55(3), 199–207.
- Tennant, N. (2021). Normalizability, cut eliminability and paradox. *Synthese*, 199(Suppl 3), S597–S616. doi: <https://doi.org/10.1007/s11229-016-1119-8>
- Tranchini, L. (2015). Harmonising harmony. *The Review of Symbolic Logic*, 8(3), 411–423. doi: <https://doi.org/10.1017/S1755020315000179>
- Tranchini, L. (2016). Proof-theoretic semantics, paradoxes and the distinction between sense and denotation. *Journal of Logic and Computation*, 26(2), 495–512.
- Troelstra, A., & Schwichtenberg, H. (2000). *Basic proof theory* (2nd ed.). Cambridge: Cambridge University Press.
- von Plato, J. (2000). A problem of normal form in natural deduction. *Mathematical Logic Quarterly*, 46(1), 121–124.
- Wansing, H. (1993). *The logic of information structures*. Berlin and Heidelberg: Springer.
- Wansing, H. (2006). Connectives stranger than tonk. *Journal of Philosophical Logic*, 35, 653–660. doi: <https://doi.org/10.1007/s10992-006-9025-z>
- Wansing, H. (in press). A note on synonymy in proof-theoretic semantics. In T. Piecha & K. Wehmeier (Eds.), *Peter Schroeder-Heister on Proof-Theoretic Semantics. Outstanding contributions to logic*. Springer.