Alte Paper lesbar machen mit pkfix

Leah Neukirchen

<leah@vuxu.org>

BayTEX 2024, Neu-Ulm

2024-09-07

Alternative Untertitel

- Probleme lösen, von denen man nicht wusste, dass man sie hat.
- 30 Jahre Fontgeschichte
- …

Wir finden ein Dokument online …

Extensional concepts in intensional type theory

Martin Hofmann

Doctor of Philosophy
University of Edinburgh
1995

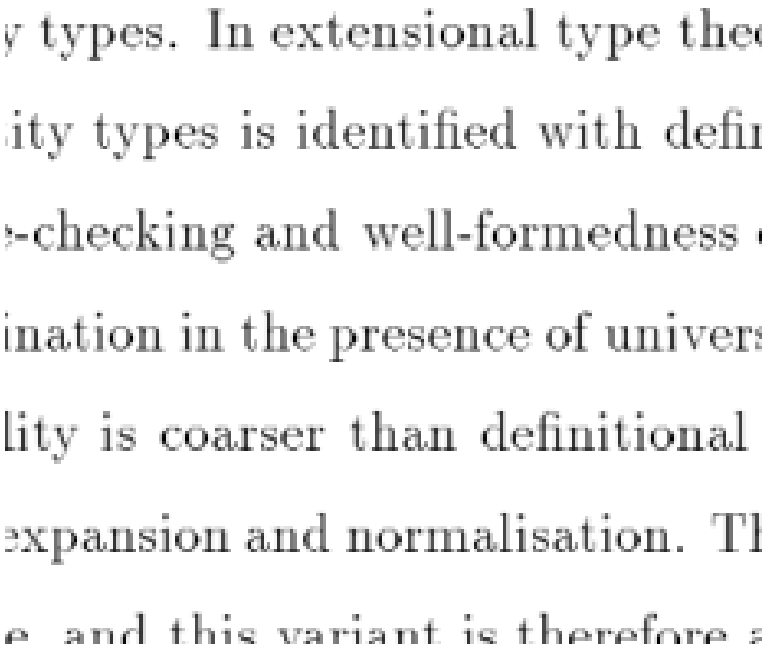
Abstract

Theories of dependent types have been proposed as a foundation of constructive mathematics and as a framework in which to construct certified programs. In these applications an important role is played by identity types which internalise equality and therefore are essential for accommodating proofs and programs in the same formal system.

This thesis attempts to reconcile the two different ways that type theories deal with identity types. In extensional type theory the propositional equality induced by the identity types is identified with definitional equality, i.e. conversion. This renders type-checking and well-formedness of propositions undecidable and leads to non-termination in the presence of universes. In intensional type theory propositional equality is coarser than definitional equality, the latter being confined to definitional expansion and normalisation. Then type-checking and well-formedness are decidable, and this variant is therefore adopted by most implementations.

However, the identity type in intensional type theory is not powerful enough for formalisation of mathematics and program development. Notably, it does not identify pointwise equal functions (functional extensionality) and provides no means of redefining equality on a type as a given relation, i.e. quotient types. We call such capabilities *extensional concepts*. Other extensional concepts of interest are uniqueness of proofs and more specifically of equality proofs, subset types, and propositional extensionality—the identification of equivalent propositions. In this work we investigate to what extent these extensional concepts may be added to intensional type theory without sacrificing decidability and existence of canonical forms. The method we use is the translation of identity types into equivalence relations defined by induction on the type structure. In this way type theory with

Bei genauerer Betrachtung


y types. In extensional type the
ity types is identified with defi
-checking and well-formedness
ination in the presence of univer
lity is coarser than definitional
expansion and normalisation. Tl
e, and this variant is therefore

Es werden Bitmap-Fonts verwendet, die gedruckt okay aussehen, am Bildschirm aber eher unangenehm sind.

Eine kurze Geschichte der TEX-Schriftformate

Ursprünglich interessiert sich TEX nicht für Schriftarten, sondern nur für ihre Metriken. Im DVI-Format steht nur, welcher Buchstabe an welche Position gesetzt werden soll.

Für TEX 82:

- TEX Font Metrics, `.tfm`
- Virtual fonts, `.vf`

Woher kommt die Schrift selbst?

- Bitmap fonts[1], erzeugt von METAFONT: `.gf` → `.pk`
- PostScript Type 1 (Outline): `.pfa`, `.pfb`
- PostScript Type 3 (PostScript)
- TrueType/OpenType: `.ttf`, `.otf` (seit XeTeX, LuaTeX)

Schriften vom Type 3 können Bitmap-Daten enthalten!

Alte Versionen von `dvips` erzeugten Type 3 aus den METAFONT-Bitmaps.

(Bekommt man auch heute noch hin, mit Gewalt: `dvips -V1`)

[1] Können für den Drucker optimiert sein!

Problem erkannt, was nun?

Heiko Oberdiek hatte die großartige Idee, ein Programm zu schreiben, dass

- Type 3 Bitmap-Fonts von dvips erkennt und
- durch Type 1 Outline-Fonts ersetzt.

Dieses Programm heißt pkfix und ist auf dem CTAN bzw. in TeXLive zu finden.

```
% pkfix hello.ps hello-fix.ps
PKFIX 1.7, 2012/04/18 - Copyright (c) 2001, 2005, 2007, 2009, 2011, 2012 by Heiko Oberdiek.
*** Font conversion: `cmr10' -> `CMR10'.
==> 1 converted font.
```

Die Datei hello-fix.ps enthält nun den entsprechenden Type 1-Font, und da die Schriftmetriken übereinstimmen, sieht das Ergebnis auch gut aus.

Einen Schritt zurück

```
% pkfix ECS-LFCS-95-327.ps ECS-LFCS-95-327-fix.ps
PKFIX 1.7, 2012/04/18 - Copyright (c) 2001, 2005, 2007, 2009, 2011, 2012 by Heiko Oberdiek.
!!! Warning: dvips version 5.495 does not generate the required font comments!
!!! Error: Missing comment `%DVIPSParameters'!
```

Leider wurde die Postscript-Datei aus dem Ausgangsproblem mit einer sehr alten Version von dvips erzeugt, die nicht als Kommentar angibt, welche Schriften verwendet werden.

(Immerhin gibt es die dvips-Version an, noch ältere Dokumente tun nicht mal das …)

Zeit für Heuristik

Das vorliegende Problem lässt sich aus Mangel an Informationen nicht mehr exakt lösen, daher muss man nun anfangen, zu raten.

Das beiliegende Programm `pkfix-helper` versucht, anhand der Schriftmetriken zu erraten, um welche Computer Modern-Schriften es sich handelt! Dazu werden die existierenden TFM-Metriken mit denen aus dem Type-3-Font verglichen:

```
% pkfix-helper ECS-LFCS-95-327.ps ECS-LFCS-95-327-helper.ps
Reading ECS-LFCS-95-327.ps ... done.
Bitmapped fonts are typeset at 300 DPI.
Determined the page size to be 598.235pt by 845.157pt (A4).
Total number of Type 3 fonts encountered: 41
Finding character widths ... done.
Reading TFM files ... done (103 TFMs in 293 scaling variations).
```

```
Matching fonts:
    Processing FL ... done (cmr12 @ 1X with mismatch=0.72376, tied with cmsl12 @ 1X).
    Processing FB ... done (cmr10 @ 1.095X with mismatch=0.37143, tied with cmsl10 @
1.095X).
    Processing FK ... done (cmti12 @ 1X with mismatch=0.28691).
    Processing FH ... done (cmmi12 @ 1X with mismatch=0.43607).
    Processing FI ... done (cmbx12 @ 1X with mismatch=0.33921).
    Processing Fg ... done (cmtt10 @ 1.095X with mismatch=0.06037, tied with cmitt10 @
1.095X, cmsltt10 @ 1.095X, and cmtcsc10 @ 1.095X).
...
```

Man beachte, dass die Metriken nicht eindeutig sind, insbesondere bei den Monospace-Varianten. Auch wurden Schriften gerne skaliert, da CM nur einige Punktgrößen hat.

(Die CM-Fonts sind seit 1992 unverändert.)

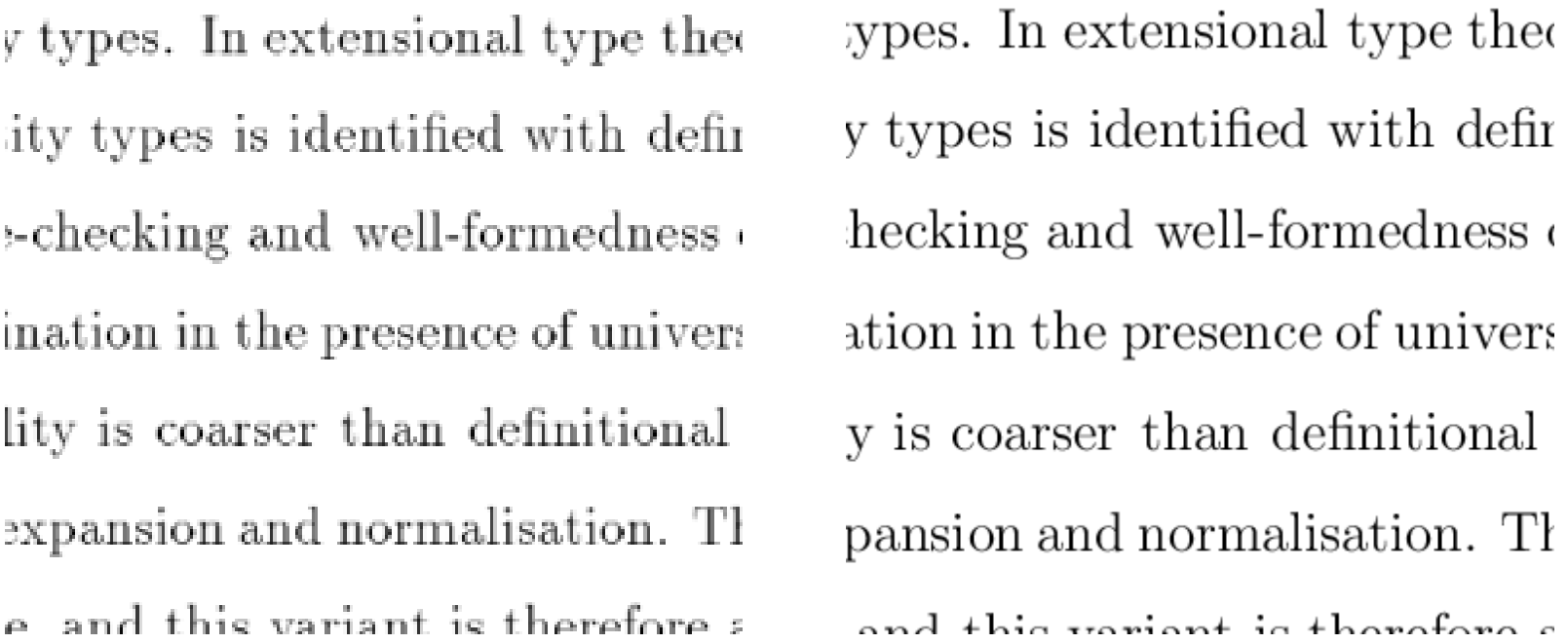
Die Ausgabe von `pkfix-helper` hat dann die benötigten Kommentare:

```
...
%DVIPSBitmapFont: Fa cmbx12 24.88 1
/Fa 1 84 df<00007FF00003000007FFFF000700001FFFFFE00F00007FFFFFF01F0000
...
```

Diese Ausgabe kann man nun wieder in `pkfix` stecken:

```
% pkfix ECS-LFCS-95-327-helper.ps ECS-LFCS-95-327-fix.ps
PKFIX 1.7, 2012/04/18 - Copyright (c) 2001, 2005, 2007, 2009, 2011, 2012 by Heiko Oberdiek.
*** Font conversion: `cmbx12' -> `CMBX12'.
*** Font conversion: `cmex10' -> `CMEX10'.
*** Font conversion: `cmcsc10' -> `CMCSC10'.
...
==> 41 converted fonts.
==> 7 merged fonts.
```

Ein optischer Vergleich zeigt, dass das gut funktioniert hat.



Mit ps2pdf können wir ein gut lesbares PDF erzeugen.

Lehren

- Archivformate wollen wohlüberlegt sein
  - Einfacher gesagt als getan, da viele Daten produziert werden, bevor klar ist, dass man gerade ein Archiv anlegt.
- Metadaten sind wertvoll
- verlustbehaftete Datenformate und Kompression sind zu vermeiden
- ein modernerer Ansatz wäre, die Schriften anhand ihres Bilds zu vergleichen

Nebenbei: Man findet von der Datei auch eine Version als `.dvi`, dann reicht es, einfach `dvipdfm` zu verwenden… allerdings findet man in der Praxis eher `.ps`.

Vielen Dank!

Noch Fragen?