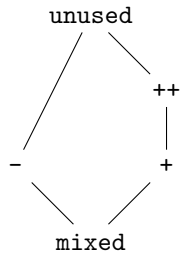
Read the ~~mood~~mode and stay positive

Lucas Escot[1], Josselin Poiret[2], Joris Ceulemans[3], Andreas Nuyts[3] and Malin Altenmüller[4]

[1] TU Delft, Netherlands
[2] ENS de Lyon, France
[3] imec-DistriNet, KU Leuven, Belgium
[4] University of Strathclyde, Scotland

Languages like Coq and Agda forbid users to define non-strictly-positive data types [AAG05]. Indeed, one could otherwise very easily define non-terminating programs. However, this strict-positivity criterion is nothing more than a syntactic restriction, which prevents sometimes perfectly reasonable and innocuous data types to be defined. We present ongoing work on making positivity checking more modular in Agda, by allowing *polarity annotations* in function types and making it possible to enforce the variance of functions simply by type checking.

The polarity modal system. We introduce a new *modal system* [GKNB21] aptly called the *polarity* [AM04, Abe06] modal system. It consists of a partially-ordered set made out of five *polarities*: `unused`, `-`, `+`, `++` and `mixed`. These elements correspond to the permitted uses of bound variables. The polarities are given the partial order shown on the right, to be read from bottom to top (e.g. `+` $\leq$ `++`) as going from less restrictive to more restrictive: a variable bound with the `mixed` polarity is allowed to appear anywhere; a variable bound with polarity `++` can only appear to the right of arrows; variables bound with polarity `-` (resp. `+`) can appear to the left of an odd (resp. even) number of arrows; and a variable bound with the `unused` polarity can only be used to define constant functions (much like irrelevance). This modal system is given a composition operation $\circ$ whose table we write below:



$\circ$	`mixed`	`+`	`++`	`-`	`unused`
`mixed`	`mixed`	`mixed`	`mixed`	`mixed`	`unused`
`+`	`mixed`	`+`	`+`	`-`	`unused`
`++`	`mixed`	`+`	`++`	`-`	`unused`
`-`	`mixed`	`-`	`-`	`+`	`unused`
`unused`	`unused`	`unused`	`unused`	`unused`	`unused`

$a \circ b$ is to be understood as the *most restrictive polarity* a variable can be bound with, such that it can be used with polarity a in a term that is itself used with polarity b. `unused` is the absorbing element and `++` is the neutral element of this operation. It gives rise to a (left) *division* operation $\backslash$, defined such that $\mu \leq \delta \circ \nu \iff \delta \backslash \mu \leq \nu$ for any δ, μ, ν. The operations $\circ$ and $\backslash$ form a Galois connection.

Typing rules. After attributing a polarity to every variable in context and function domains (`@r` $x : A$), and extending the left-division operation to contexts (${}_{\mathtt{r}\backslash}\Gamma$) variablewise, we introduce the typing rules of the polarity modal system. We implicitly use Russel-style universes. Note that the context of the premise of T-El is left divided by `unused`, which is equivalent to changing all the annotations in the context to `mixed`: informally, variable use in type judgements does not matter.

$$\frac{{}_{\mathtt{unused}\backslash}\Gamma \vdash A : \mathtt{Set}}{\Gamma \vdash A \text{ type}}\ \textsc{t-El} \qquad \frac{\Gamma \text{ ctx} \qquad \Gamma \vdash A \text{ type}}{\Gamma, \mathtt{@r}\ x : A \text{ ctx}}\ \textsc{ctx-Ext} \qquad \frac{\mathtt{@r}\ x : A \in \Gamma \qquad \mathtt{r} \leq \mathtt{++}}{\Gamma \vdash x : A}\ \textsc{t-Var}$$

Modal function types (T-Pi) use their domain negatively[1]. Note that left-dividing by `-` corresponds to inverting the annotation's polarity, except for `++` which becomes `unused`. Regardless of the annotation `@r` on the variable x, x is bound in the codomain as `mixed` (the weakest modality). This is in line with T-El: since usage at the type level does not count, we do not constrain it.

$$\frac{_{\text{-}\backslash}\Gamma \vdash A : \texttt{Set} \qquad \Gamma, \texttt{@mixed}\ x : A \vdash B : \texttt{Set}}{\Gamma \vdash (\texttt{@r}\ x : A) \to B : \texttt{Set}}\ \textsc{t-Pi} \qquad \frac{\Gamma, \texttt{@r}\ x : A \vdash t : B}{\Gamma \vdash \lambda x.t : (\texttt{@r}\ x : A) \to B}\ \textsc{t-Lam}$$

$$\frac{\Gamma \vdash u : (\texttt{@r}\ x : A) \to B \qquad _{\texttt{r}\backslash}\Gamma \vdash v : A}{\Gamma \vdash u\ v : B[v/x]}\ \textsc{t-App}$$

And the monad was free, the (fix)point taken. We implemented the new modal system and the typing rules presented above in Agda[2]. Since other modal systems were readily available — erasure [Dan19, ADV21], irrelevance[3], cohesion [LOPS18] — a lot of the infrastructure was in place. Still, our work highlighted some deficiencies in the current implementation of modalities [NPE+23]. Using our modified version of Agda, the following annotations are valid and taken into account by the type checker.

```
F : @++ Set → Set          G : @- Set → Set          H : @+ Set → Set
F X = Nat → X              G X = X → Nat             H X = (X → Nat) → Nat
```

Above, only `F` is *strictly* positive and can be annotated as such without the type checker getting in the way. We extended Agda's positivity checker so that it also uses the polarity of functions during the analysis, allowing the definitions of both the well-known free monad construction `Free` and the least fixed point `Mu` of any strictly positive functor. Note that `F` and `A` could themselves be annotated `++`, we refer to the pull request for more elaborate examples.

```
data Free (F : @++ Set → Set)          data Mu (F : @++ Set → Set) : Set where
    (A : Set) : Set where                In : F (Mu F) → Mu F
  Pure : A            → Free F A
  Free : F (Free F A) → Free F A
```

Next steps. This work is ongoing and much is left to be done. On the semantics side of things, a model for the polarity modalities is still eluding us, especially for the `++` polarity, and it will most likely require looking deeper at directed type theory. The usefulness of our annotations is hindered by the lack of subtyping in Agda, preventing one to use functions of type `@++ Set → Set` wherever `Set → Set` is expected. Even if manual eta-expansion appeases the type checker, a user of our annotation system has to redefine all the usual constructions from scratch. A further complication is the fact that polarity and subtyping can interact in a non-trivial way [Abe06]. Another question left to be answered is whether it is safe to add the primitive `fmap : (F : @+ Set → Set) (f : A → B) → F A → F B` to Agda such that it knows `fmap` is terminating and always reduces as expected. While relaxing the strict-positivity criterion to simply positive data types has been shown to be inconsistent in presence of an impredicative sort [CP88], one can wonder whether it would be safe in Agda [BMS18]. We also want to investigate whether our polarity system could replace Agda's positivity checker entirely, greatly simplifying the implementation and perhaps even improving type checking performance.

[1] As remarked by Nuyts [Nuy15, eqs. (2.43, 2.58 modulo typo)], more care is needed if one wants to take higher categorical structure into account.

[2] https://github.com/agda/agda/pull/6385

[3] The Agda implementation does not seem to follow any specific literature for irrelevance.

Acknowledgements Joris Ceulemans and Andreas Nuyts hold a Phd Fellowship and a Postdoctoral Fellowship (resp.) from the Research Foundation - Flanders (FWO). This research is partially funded by the Research Fund KU Leuven.

We would like to thank the TYPES 2023 reviewers for their comments and suggestions.

References

[AAG05] Michael Gordon Abbott, Thorsten Altenkirch, and Neil Ghani. Containers: Constructing strictly positive types. *Theor. Comput. Sci.*, 342(1):3–27, 2005. doi:10.1016/j.tcs.2005.06.002.

[Abe06] Andreas Abel. Polarized Subtyping for Sized Types. In Dima Grigoriev, John Harrison, and Edward A. Hirsch, editors, *Computer Science – Theory and Applications*, Lecture Notes in Computer Science, pages 381–392. Springer, 2006. doi:10.1007/11753728_39.

[ADV21] Andreas Abel, Nils Anders Danielsson, and Andrea Vezzosi. Compiling programs with erased univalence. 2021. Draft. URL: https://www.cse.chalmers.se/~nad/publications/abel-danielsson-vezzosi-erased-univalence.pdf.

[AM04] Andreas Abel and Ralph Matthes. Fixed points of type constructors and primitive recursion. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic, 18th International Workshop, CSL 2004, 13th Annual Conference of the EACSL, Karpacz, Poland, September 20-24, 2004, Proceedings*, volume 3210 of *Lecture Notes in Computer Science*, pages 190–204. Springer, 2004. doi:10.1007/978-3-540-30124-0_17.

[BMS18] Ulrich Berger, Ralph Matthes, and Anton Setzer. Martin Hofmann's case for non-strictly positive data types. In Peter Dybjer, José Espírito Santo, and Luís Pinto, editors, *24th International Conference on Types for Proofs and Programs, TYPES 2018, June 18-21, 2018, Braga, Portugal*, volume 130 of *LIPIcs*, pages 1:1–1:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.TYPES.2018.1.

[CP88] Thierry Coquand and Christine Paulin. Inductively defined types. In Per Martin-Löf and Grigori Mints, editors, *COLOG-88, International Conference on Computer Logic, Tallinn, USSR, December 1988, Proceedings*, volume 417 of *Lecture Notes in Computer Science*, pages 50–66. Springer, 1988. doi:10.1007/3-540-52335-9_47.

[Dan19] Nils Anders Danielsson. Logical properties of a modality for erasure. 2019. Draft. URL: https://www.cse.chalmers.se/~nad/publications/danielsson-erased.pdf.

[GKNB21] Daniel Gratzer, G. A. Kavvos, Andreas Nuyts, and Lars Birkedal. Multimodal dependent type theory. *Log. Methods Comput. Sci.*, 17(3), 2021. doi:10.46298/lmcs-17(3:11)2021.

[LOPS18] Daniel R. Licata, Ian Orton, Andrew M. Pitts, and Bas Spitters. Internal universes in models of homotopy type theory. In Hélène Kirchner, editor, *3rd International Conference on Formal Structures for Computation and Deduction, FSCD 2018, July 9-12, 2018, Oxford, UK*, volume 108 of *LIPIcs*, pages 22:1–22:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSCD.2018.22.

[NPE+23] Andreas Nuyts, Josselin Poiret, Lucas Escot, Joris Ceulemans, and Malin Altenmüller. Proposal: Multimode Agda, 2023. URL: https://github.com/anuyts/public/blob/1edae3589f9db6f0757699b33f960b1db061e7a4/agda/modal.md.

[Nuy15] Andreas Nuyts. Towards a directed homotopy type theory based on 4 kinds of variance. Master's thesis, KU Leuven, Belgium, 2015.