Revision Log

September 25, 2018

1 README

Replies written to reviewers are in Reply: green. Polish anything in the paper/reply as you see appropriate.

2 Associate Editor

From Associate Editor Jeffrey Foster:

Recommendation #1: Major Revision

Associate Editor

Comments for Author:

Thank you for your submission to TOPLAS. The reviewers all feel this is a strong, well-written paper that makes important contributions. However, they do point out a range of issues that should be addressed before the paper can be accepted. Hence I am recommending you complete a Major Revision of this work.

In preparing your revision, please be sure to take all of the reviewers' detailed comments into account. Please be especially sure to address the key concerns, including Reviewer 1's comments about Section 4.4 and about being more clear in acknowledging the relationship of the proof to prior work. Also be sure to address Reviewer 2's comment about one of the examples.

3 Referee: 1

3.1 Summary

The paper presents new results in gradual typing, with one contribution being consistent subtyping for implicit polymorphism, another being type inference

for gradual typing with higher-rank polymorphism. Some results depend on a conjecture, but even if the conjecture turns out to be false, the work done under that assumption is likely to be valuable.

To address the main TOPLAS criteria:

1. This is one of the best papers I have seen in the last year. Either the first half of the paper or the second might be worth consideration alone, but the combination is certainly substantial.
2. Gradual typing is "hot", and the specific research direction is well-motivated.
3. The presentation is largely effective. There is a substantial inaccuracy in section 4.4, some inconsistent notation, and at least one missing description of notation. Moreover, the paper needs to clarify that some of the manual proofs (in the appendix) are adapted (sometimes with no real differences) from a previous paper; that previous paper is amply cited as inspiration, but the authors need to point out the extent to which they have relied on others' work for both the structure of the metatheory (statements of lemmas and the lemmas' names) and some of the proofs themselves.

The remainder of my review is structured as follows:

1. some thoughts on section 4.4;
2. relatively major issues that need to be addressed;
3. minor comments (where the authors need not make the changes if they prefer the current version).

3.2 Thoughts on section 4.4

In Jafery and Dunfield 2017, A $\sqsubseteq$ B means A is more precise (or "less imprecise") than B, not less. Section 4.4 flips the meaning; the paper should point this out (or make the notation consistent).

While it is true that their definition of directed consistency *looks* somewhat like the submission's Definition 4.1, the directions don't match up. 4.1 says you can move to a supertype (which, given a reasonable subtyping relation, is always safe: it should never immediately cause a risky cast), then "blur" the type via consistency, then move to a supertype. Jafery and Dunfield's directed consistency says you can gain precision (which might fail), move to a supertype, and then lose precision.

Maybe the reference to 4.1 at p. 14 / line 26 was meant to be a reference to 4.4 (consistent subtyping in AGT)?

Even if so, the discussion in the next paragraph (27-31) is clearly about 4.1. While it is reasonable to ask which relation is "more fundamental", that has

little to do with the superficial fact that both directed consistency and 4.1 "look[] very similar".

At 40-41 the paper says "It may seem that precision is a more atomic relation..." I agree. But then the paper claims (okay, implies, but I am not sure how else to read the last sentence "Thus precision can be difficult to extend...") that, because it is (I would say, has often been) easier to define consistency exnihilo than precision, precision is really less atomic. That doesn't follow.

Reply:

1. We added a note to clarify that the definition of directed consistency is adapted to match our interpretation of the notation $\sqsubseteq$, for the sake of consistency with Cimini and Siek 2016 throughout the paper. That is, $A \sqsubseteq B$ means A is less precise than B. This definition of $\sqsubseteq$ was given in appendix, and is now moved to the main text (see Fig. 10).
2. We deleted "two definitions look similar". We agree with the reviewer that this superficial fact is misleading.
3. We rephrased the sentence of the reference to 4.1 that was at p. 14 / line 26 to make it more clear. As the definition of directed consistency coincides with Definition 4.4 (Consistent Subtyping in AGT), by Corollary 4.5 (Eqivalence to AGT on Simple Types), the definition coincides with our definition of consistent subtyping, i.e. Definition 4.1 (Consistent Subtyping).
4. Indeed the question ("consistency or precision is more atomic") cannot be easily answered without principal analysis. We rephrased the paragraph to focus more on the discussion about their relation, rather than giving an certain answer to this question.

I see two research questions here.

The first question is "moral": in a gradual language, what relation(s) "should" come first? That is a conceptual question.

The second is a pragmatic question: in a gradual language, what relation(s) should come first to make the designer's life easier?

Ideally, the questions would have the same answer, but there is no particular reason to believe they do. Lines 40-48 are maybe an answer to the second question, but the phrase "more atomic" connotes (to me) the first question.

Reply: We don't have a general answer to this question. As far as we are concerned, it is easier for type consistency to come first and to make the designer's life easier, for exactly the reason we stated in the paragraph. though we cannot claim it for sure without a deeper analysis.

3.3 Relatively major issues

11, footnote 5: how is your simplified version different?

Reply: We extended the footnote with more explanations. This simplified version is presented as one of the key ideas of the design of type consistency, which is later amended with labels.

13, footnote 6: at least in this section, there is only one place that $\rightarrow$ is used as a set-level operator (in Def. 4.3); it would be more clear to write out

$$\gamma(A \rightarrow B) = \{S_A \rightarrow S_B | S_A \in \gamma(A) \quad \text{and} \quad S_B \in \gamma(B)\}$$

As it is, the reader has to find the definition of lifted operators in the AGT paper, and figure out that you are writing $\rightarrow$ instead of $\tilde{\rightarrow}$.

Reply: We made the change as suggested.

In Def. 4.4 it would be good to mention that the A_1, B_1 in the concretizations are *static* types (this is explicit in the AGT paper, because it uses different meta-variables), though this is implicit in your explanation of concretization.

Reply: We made the change as suggested.

14, lines 22-23: "In their setting, precision is defined for type constructors and subtyping [is defined] for static types." The first part is slightly misleading, and the second part is not accurate:

1. Jafery and Dunfield define precision for type constructors, and then straightforwardly lift it to types.

Reply: We clarified the sentence as suggested.

2. They define subtyping over *all* gradual types. Subtyping over sums (which are the only gradual types in their system) is defined using subtyping over "sum constructors". Arguably, their system mixes up precision and subtyping, because the subtyping relation is not defined purely over static types. It seems likely that could be disentangled, leading to a directed consistency relation that has a static relation (subtyping over static types, not gradual types) bracketed by (im)precision, but that's not what they did.

Reply: We clarified the sentence as suggested.

Section 5.3 appears to mix up (opposite) meanings of the symbol $\sqsubseteq$:

Meaning 1: O the same program except that e2 has more unknown types". This is the reading of $\sqsubseteq$ as "more precise than" or "less imprecise than": e1 is more precise, e2 is less precise (because it has more unknown types).

Meaning 2: Within Lemma 4 and Definition 5.4, this is reversed. In Definition 5.4, e' $\sqsubseteq$ e means that e is more precise. If the "gradually typed program evaluates to a value" (that value being v), then *less* precise annotations (those on e') translate to s', which evaluates to v', where v' is "less precise".

My personal preference is strongly for Meaning 1, which is consistent with the original statement of the gradual guarantee (Siek et al. 2015). The paper should at least be internally consistent.

Reply: This is indeed a typo. It should be "e1 $\sqsubseteq$ e2 means that e1 and e2 are the same program except that e1 has more unknown types". That is, Meaning 2. This interpretation matches our definition of $\sqsubseteq$, which was in the appendix and is now given in Fig 10.

In 6.0, various notations are explained, including one-hole notation for contexts, but the two-hole notation $\Gamma[\theta_1][\theta_2]$ is not explained—even though at 23 line 40, the notation is used with "Recall that...".

Reply: We added the missing explanation.

In Def. 9.11, the case for $(\Gamma, a_S)_A$ where a_S does *not* occur in A is missing. Should it be the following?

$$(\Gamma, \widehat{a}_S)\ \Gamma_A, \widehat{a}_S \quad \text{if } \widehat{a}_S \text{ occurs in } A$$

Reply: Yes. We added the missing case.

Arguing for the decidability of the algorithmic system (page 25) only by citing Dunfield and Krishnaswami (2013) is somewhat unsatisfying: since decidability is not obvious, it would be better to show decidability outright for the actual system at hand. If their proof strategy continues to work, doing the proof should be easy. (If this proof was mechanized, then just mention that. But my impression is that your proofs in this section were not mechanized.)

Reply: We added detailed proofs showing decidability of our extended algorithmic system in appendix C. We significantly expanded section 6.4 to highlight the key lemmas of decidability. Due to our gradual setting, we have a revised measure where we need to count the number of unknown types when proving decidability of consistent subtyping. This is a key difference from the original decidability proofs in the DK system. The adoption of matching judgment also simply the reasoning a bit when proving decidability of typing.

32 line 44: "the dynamic gradual guarantee is a story of the good choices": Please explain what you mean by a "story of the good choices". I don't know what a "story of choices" would be.

Reply: We rephrased the sentence to make it clear. "*...so we can restrict the discussion of dynamic gradual guarantee to representative translations.*"

37 line 8: "principle" should be "principal".

Reply: Fixed.

37 lines 32-34: Saying that no previous approaches "can instruct us how to define..." is very strong; it would be better to say they did not instruct you,

or that they do not directly lead to a definition of consistent subtyping for polymorphism.

Reply: We weakened the argument as suggested. "*...none of these approaches instructed us how to define consistent subtyping for polymorphic types.*"

38 line 23: Remind the reader what Siek and Vachharajani's observation was.

Reply: We added explanations to make the description more clear.

38 line 34: "a sole inspiration": "Sole" means "only", so "a" does not make sense; did you mean "the sole inspiration"? But it was probably not the sole inspiration, rather the main or primary inspiration. Unless you meant that the declarative system is the sole inspiration for the algorithmic system, in which case you should rephrase (and say "basis" rather than "inspiration").

Reply: We changed it to "*...the main inspiration...*"

Appendix:

Much of the infrastructure (at least for these manual proofs for the relevant part of the paper) is identical to Dunfield and Krishnaswami (2013), up to and including the names of many lemmas (even their odder names, such as "Softness Goes Away" and "Confluence of Completeness"). This is good evidence that their proofs can feasibly be adapted and extended (unlike many papers that don't give the metatheory in enough detail), but the paper does not make the extent of this reuse clear (I don't think it was mentioned at all). In at least some cases, the proofs are repeated almost verbatim, so appropriate credit is mandatory.

Reply: We added credits to DK in appendix E.1: basically all the syntactic properties of context extensions are copied from theirs verbatim because we have the same context extension rules. However, the rest of lemmas and theorems are provided with detailed proofs, with necessary changes to support our new gradual type system.

3.4 Minor comments

In the abstract, including citation years would be helpful.

Reply: It is our personal preference to avoid citation years in the abstract. We slightly revised the abstraction by adding more texts to disambiguate when there can be confusion.

The phrase "armed with a coherence conjecture" is odd, since (being a conjecture) it is not an effective weapon. "Assuming a coherence conjecture"?

Reply: We made the change as suggested.

Quotations (e.g. p. 2 line 10-11) should not be in italics.

Reply: Fixed.

p. 2 line 41: "same flavor" should be "some flavor"

Reply: Fixed.

p. 3: Footnote 2 should go in the main text.

Reply: Thanks for the advice. We state it explicitly in the main text now. *"The second goal of this paper is to present the design of GPC, which stands for Gradually Polymorphic Calculus..."*.

p. 9: "dispersed with": "littered with" (or say that "toDyn and fromDyn are dispersed throughout the code")

Reply: Fixed.

p. 10 line 41: "Importantly" is redundant with "crucial".

Reply: Fixed.

p. 10, p. 13: "so-called" has a connotation that the name given is not good (e.g. writing "so-called *naive subtyping*" if I think the term "naive subtyping" is misleading). Using italics for the term itself, without "so-called", is sufficient.

Reply: Fixed.

p. 19 line 45: "a static program that is typeable...if and only if...": delete "that".

Reply: Fixed.

Fig. 13: The "Instantiation I" and "Instantiation II" judgments are not distinguished by syntax:

$$\Gamma \vdash \widehat{a} \lessapprox \widehat{a} \dashv \Delta$$

could mean either "Instantiation I" with $\widehat{a}$ as the right-hand A, or "Instantiation II" with $\widehat{a}$ as the left-hand A. It's not actually broken, because AS-EVAR handles the situation when it would otherwise arise "from" Fig. 12, and within Fig. 13 it is clear (e.g. in INSTR-ARR the first premise is clearly switching to "Instantiation I" because `A_1` is on the right). But it is potentially confusing. Even if the authors do not want to change the notation, this issue should be mentioned.

Reply: The above judgment does not hold, because both in INSTL-SOLVE and INSTR-SOLVE, it requires τ to be well-typed in the context before $\widehat{a}$, which is impossible for this case. And exactly, this case is handled in AS-EVAR.

20 line 43: "henceforth DK system": the abbreviation is not actually used later.

Reply: We replaced some of the citations with DK system to reduce repetitive citations.

22 lines 45-46: Could mention that as-forallR is invertible, so in an implementation the choice is easy.

Reply: We made the change as suggested.

23 line 30: "Two twin judgments" is redundant: "twin" implies two. I would say "Two symmetric judgments".

Reply: Fixed.

In 6.2, it seems worth mentioning that the idea implemented by INST(L,R)-SOLVE is due to Cardelli (see citation in Dunfield and Krishnaswami 2013).

Reply: We added citation to Cardelli 1993 in Section 6.2.

29 line 20: hyphen in "syntax directed", line 21: "combing" should be "combining"

Reply: Fixed.

32 Def. 9.7: the last `\Rightarrow` should probably be `\Longrightarrow`,

Reply: Fixed.

but I would find the definition easier to parse in English: "if ... then for all C such that ...implies ...". Mixing object- and meta-level symbols (like ∀) is confusing.

Reply: We replaced object-level ∀ with the text "for all" in Definition 5.1 and 9.7 to make it more clear.

34 line 24: add space before "to"

Reply: Fixed.

The notation `$\Gamma_{\mathbb C}$` is maybe too lightweight: it looks like a meta-variable. Using a superscript would be somewhat better, but I would prefer a more function-like notation `("contaminate(\Gamma, C)"`?

Reply: We made the change as suggested.

34 line 45: Emphasize that substituting in the input contexts is not just unfortunate, but would effectively turn the input contexts into something else entirely: the point of the input context is that it is input, with necessary changes (like solving existentials) appearing in the output context.

Reply: We added this clarification accordingly.

34 line 49: "every static existential variables": change "every" to "all"

Reply: Fixed.

36 line 21: "with static" should be "with a static"

Reply: Fixed.

36 line 44: "later" should be "latter"

Reply: Fixed.

38 line 36: "higher rank" needs a hyphen

Reply: Fixed.

line 38: "as such" should be "such as"

Reply: Fixed.

line 50: "insight in" should be "insight into"

Reply: Fixed.

39 line 9: "real world" needs a hyphen

Reply: Fixed.

Appendix:

E.6: "if any only if" should be "if and only if".

Reply: Fixed.

Definition F.1 is missing the case for the marker $\blacktriangleright_{\widehat{a}}$.

Reply: Fixed.

In Lemma F.2, the first `\emptyset` should be the empty context.

Reply: Fixed.

4 Referee: 2

4.1 Comments to the Author

The paper presents a gradually typed language with Curry-style (implicit) higher-rank polymorphism, focusing on syntactic aspects, especially the static "refined" criteria of gradual typing and correctness of a bidirectional type-checker in the style of Dunfield-Krishnaswami. A shorter paper has previously been presented at ESOP and the main additions in the submission seem to be sections 3, 8.2 and 9.

The contributions of the paper are strong. The authors improve over previous work considerably in the syntax of a gradual polymorphic source language: the declarative and algorithmic systems are both straightforward, a sign of a good design.

I think the most confused point in the paper is about instantiation of a forall type with a dynamic type. In the main body of the paper, adapted from the short version, the system does not allow this because the dynamic type is not a monotype. However, the last motivating example (heterogeneous containers) instantiates the polymorphic Scott-encoded 'nil' and 'cons' functions with the dynamic type. Finally, in section 9, they present a system with "static and

gradual type parameters", where an otherwise un-constrained gradual type parameter can be instantiated with the dynamic type. My understanding is that the heterogeneous container example is based on this final system, which is also the one followed by the implementation, but this needs to be explained more clearly in the text, preferably with a small representative example.

Reply: We added a note for the heterogeneous container in Section 3 to clarify that the heterogeneous list can only get the dynamic type in the extended type system presented in Section 9.

Ningning: I don't understand here about the "a small representative example". An example for what?

Finally, before my more targeted comments, a philosophical question. One of the main contributions is a new formulation of consistent subtyping that shows that the definition of Siek and Taha is incorrect. We can see in retrospect that the presentation in Siek-Taha was a coincidence based on the simplicity of their the types involved. While the new characterization is clearly an improvement, without a *semantic* criterion it's not clear why this is correct. Why should we believe *in principle* that the correct definition of consistent subtyping is "$A \lesssim B$ iff $A <: A' \sim B' <: B$" and that this will be work in the future? The authors show it also works for a top type, but it would be nice to have a more conceptual explanation.

Reply: We don't have a general answer to this question.
We don't claim that the definition of Siek and Taha is incorrect. Their definition works well in gradual typing for objects. Nor do we claim that our definition is *the* correct one. We merely claim that it generalizes and subsumes the definition from Siek and Taha, and it works for polymorphic types and Top types.
It would be interesting to investigate whether our notion of consistent subtyping has a more foundational conceptual explanation, for example, whether it would coincide with AGT on polymorphic types. We leave that as future work.

4.2 More Targeted Technical Comments

1. I suggest you downgrade the status of Corollary 5.3 to Lemma rather than corollary, because it is generally quite difficult to make these contextual equivalence results completely precise. In particular, the proposed reason for the corollary to be true ("the only role of types and casts is to emit cast errors") is a major component of the dynamic gradual guarantee.

   Reply: We made the change as suggested.

2. It is worth pointing out that even with parametricity, Scott encodings are not correct in CBV with effects, and so the use of polymorphism does not alleviate the need for algebraic datatypes from the langauge. For instance, $(\lambda n.\lambda c.\Omega)$ is a valid inhabitant of the Scott encoding but does

not correspond to any (strict) list. In a language like Haskell though you are probably fine.

Reply: We made a note as suggested.

3. Section 5.2 can be improved in its explanation of contextual approximation/equivalence. Specifically, approximation is only introduced as an intermediate notion for presenting equivalence, so it would be clearer to present the following direct characterization of equivalence, that has the benefit of being almost a direct translation into math of "equivalence up to error on either side":

```
s1 =~ s2 iff forall C : (...).
    C[s1] ->* blame \/ C[s2] ->* blame
    \/ (C[s1],C[s2] diverge) \/ (C[s1],C[s2] ->* n)
```

similarly Definition 9.7 could be removed and replaced with equivalence defined as:

```
s1 =~ s2 iff forall C : (...).
    (C[s1],C[s2] ->* blame)
    \/ (C[s1],C[s2] diverge) \/ (C[s1],C[s2] ->* n)
```

Reply: Thank you for the suggestion. We would like to stick to the current definition for personal preference, which is also the form adopted by Ahmed et al. 2017.

4. Page 14 "recall that consistency is in fact an equivalence relation lifted...": this interpretation is inherent to AGT, but you are not using the AGT approach. It's not clear at all where in the paper this should be recalled from.

Reply: Fixed. We now mentioned this observation in Section 4.1 where we define consistency.

5. Page 14 you present precision for the first time and as I'm sure the authors are aware the directionality of this relation is fraught in the literature, so please note for the reader when it is introduced that you are picking the opposite convention from most of the work you are citing, including the rule you are presenting.

Reply: We added a note for more explanations, and the full definition of precision is now in the main text (see Fig. 10)

6. The authors show that their definition of consistent subtyping is "correct" for a top type and a predicative forall type, and that the Siek-Taha definition fails. Both of these can be understood as *intersection types*: the top type is the intersection of the empty set and a forall type is an infinitary intersection. Does a similar property hold for a *binary* intersection? A discussion of this would improve 8.1.

Reply: We added a simple discussion in Section 8.1 as suggested.

7. In the statement of the dynamic gradual guarantee using representative translations, shouldn't "then for some B and r' with `e' : B ~~> r'`" be replaced with the stronger "for any B and r' with `e' : B ~~> r'`"? They are equivalent if 9.8 is correct?

   Reply: No, they are not. We require that $B \sqsubseteq A$. Yet, due to the generalization rule an expression can have infinitely many different types which don't necessarily satisfy that constraint. For example, 1 can have both types Int and $\forall a.\mathsf{Int}$. Indeed, $\bullet \dashv 1 : \mathsf{Int} \leadsto 1$ and $\bullet \dashv 1 : \forall a.\mathsf{Int} \leadsto \Lambda a.1$.

4.3 Writing/Typos

1. Page 12 observations 1 and 2 can be summarized by saying that consistent subtyping is a *bimodule* with respect to subtyping on each side, or more explicitly, a <:, <:-bimodule. Then definition 4.1 says that consistent subtyping is the least subtyping bimodule extending consistency.

   Reply: The reviewer is right, this is an accurate characterization. We added a note to point it out for readers who are potentially familiar with category theory.

2. Page 14: "at first sight their definition looks very similar (replacing $\sqsubseteq$ with <: and ...)". Watch out: this is at best a misleading oversimplification: notice that you need to flip the precision on one side. This definition says that consistent subtyping is a [=,=] bimodule, as all AGT lifted relations are. This should be fixed to avoid confusing the reader.

   Reply: We removed this misleading sentence.

3. Page 21: When introducing the bidirectional system for the first time, it would be very helpful to clearly identify the *modes* for all of the judgments when they are introduced. So for example when algorithmic consistent subtyping is introduced I wasn't sure at first if the types were inputs or outputs until I read through the definition.

   Reply: We added explanation in every algorithmic judgment form as suggested.

4. page 23 line 45 "rule rule"

   Reply: Fixed.

5. Page 30 the sentence starting "In order to account..." doesn't make grammatical sense

   Reply: Fixed.

6. Page 37 "There is not much work on integrating gradual typing with parametric polymorphism": I would not say that this is true at all, especially

if you broaden it to include the broader class of works on the dynamic enforcement of parametric polymorphism which go back as early as Morris 1973 "Types are not Sets" (pre-dating the formalization of parametricity!).

Reply: We rephrased the sentence to make it clear and included a simple discussion of Morris 1973 in related work.

7. Finally the last paragraph of page 39 "As future work,..." is so vague as to be useless. If you have something more specific to say, like specific features say it.

   Reply: We made the change as suggested.

5 Referee: 3

5.1 Comments to the Author

This paper makes an important contribution to the theory of gradual typing in showing how to combine gradual typing with implicit, higher-rank polymorphism. The paper generalizes the notion of consistent subtyping to handle polymoprhic subtyping, defines both a declarative and algorithmic version of the type system, and gives a translation to the polymorphic blame calculus. The paper's treatment of consistent subtyping is particularly nice in the way it further demonstrates the orthogonality of consistency and subtyping. The paper proves that the declarative and algorithm versions of the type system are in agreement and it proves that the system satisfies the static aspects of the refined criteria for gradual typing. The paper shows that their language satisfies the dynamic gradual guarantee provided that the polymorphic blame calculus does and that a conjecture about coherence holds. Most of the definitions and proofs were verified in Coq.

5.2 Issues to be fixed

p. 5

"To compose subtyping and consistency, Siek and Taha defined consistent subtyping (written $\lesssim$) in two equivalent ways:"

That is not the definition of consistent subtyping. That is Proposition 2 of Siek and Taha (2007). The definition, as you later admit, is based on the restriction operator. In several later points in the paper (page 11, 28) you make the same mistake, calling this the definition, which it is not. In general, the discussion of Siek and Taha (2007) should focus more on the actual definition (and the restriction operator) and less on Prop. 2. Section 4.5 does a better job in this regard.

Reply: We rewrote this subsection to make it clear. Now we state explicitly in the text that the restriction operator is the definition, which we refer to as the *algorithmic* consistent subtyping. $A \sim B <: C$ and $A <: C \sim B$ come from the proposition, and we refer to them as *declarative* consistent subtyping, as it servers as a good guideline on superimposing consistency and subtyping.

p. 16

The treatment of the dynamic type in the definition of consistent subtyping of Figure 8 is not novel. The first published version of consistent subtyping that takes this approach is due to Bierman and Abadi. (See the bibtex entry below, in particular, see section 6 about the prodFTS language.) The novelty of Figure 8 is in dealing with universal types.

```
@incollection{Bierman:2014aa,
Author = {Bierman, Gavin and Abadi, Mart{'\i}n and Torgersen, Mads},
Booktitle = {ECOOP 2014 -- Object-Oriented Programming},
Editor = {Jones, Richard},
Pages = {257-281},
Publisher = {Springer Berlin Heidelberg},
Series = {Lecture Notes in Computer Science},
Title = {Understanding {TypeScript}},
Volume = {8586},
Year = {2014}}
```

Reply: We added the discussion with this paper in related work (see the end of the paragraph titled " Gradual Typing."). Basically there are two differences between TypeScript and our system. First, as the reviewer mentioned, they don't have polymorphic types. Second, while our consistent subtyping inserts run-time casts, in TypeScript, type information is erased after compilation so there are no runtime casts, which makes runtime type errors possible.

p. 31

"therefore it is best to instantiate a with $\star$" "We interpret G as $\star$ since any monotype could possibly lead to a cast error."

It seems that the upshot is that you get similar results (or even the same?) as the inference algorithm of Siek and Vachharajani (2008). The discussion of Siek and Vachharajani (2008) in Section 10 (Related Work) needs to be updated to state the similarities/differences in the inference results. That is, answer the question: if you remove polymorphism from your system, how do the inference solutions compare to those of Siek and Vachharajani? Are they identical?

Reply: We updated Related Work to include a simple discussion. While we believe we get similar results as theirs, without rigorous proof we cannot claim it for sure. Our algorithm is inspired by Garcia and Cimini 2015, where they claim that their algorithm have principal types. In this sense, our algorithm should produce similar results as Garcia and Cimini 2015, which is at least as general a type as that of Siek and Vachharaiani 2008 in inferring simple types.

Ningning: Please double check here.

5.3 Minor comments and edits

p. 2

"As Siek and Taha [2007] put it this shows that"

==> insert a comma

"As Siek and Taha [2007] put it, this shows that"

Reply: Fixed.

p. 3

"and does not require the ad-hoc restriction operator of Siek and Taha [2007]"

What makes the restriction operator "ad-hoc"? On what basis do you make this derogatory remark?

Reply: We removed this remark.

p. 6

In section 2.2 it would be helpful to say something about the two subtyping rules for universal types.

Reply: We added explanations for those subtyping rules.

p. 9

"The drawback is that the code is dispersed with toDyn and fromDyn, which obscure the program logic"

==> change "disperse" to "littered" to avoid non-idiomatic phrase "The drawback is that the code is littered with toDyn and fromDyn, which obscure the program logic"

Reply: Fixed.

p. 25

"the proof strategy employed by them for decidability can be easily adapted to our setting"

==> "can be" is weak and vague "we easily adapted the proof strategy employed by them for decidability to our setting"

Reply: We added detailed proofs showing decidability of our extended algorithmic system (Appendix C). We refer to our relevant reply to reviewer 1.

p. 28

The "naive design" is painfully naive. Furthermore, the V-App1 rule would be wrong regardless of the V-Sub rule. I recommend deleting this part.

Reply: We have decided to keep this design because we were explicitly asked by a reviewer of our ESOP version if it would work. This shows that this design is not so obvious for everyone.
We slightly revised the design's explanation to emphasize from the beginning that it is wrong.

p. 29

In the proper declarative design, why not using matching and just one application rule, following more modern presentation of gradual typing?

Reply: We made the change as suggested. We updated the Coq proof as well.

p. 39

"have the dynamic guarantee"

==> "have the dynamic gradual guarantee"

Reply: Fixed.