A Kripke Logical Relation for Affine Functions:
The Story of a Free Theorem in the Presence of Non-termination

Phillip Mates
Northeastern University

Amal Ahmed
Northeastern University

June 14, 2013

Abstract

In the course of extending Ahmed and Blume's recent work on fully abstract CPS translation to a language with non-termination, we have encountered the need for a free theorem that cannot be proved using existing logical relations. The free theorem involves a mix of affine functions and *affinely-treated* intuitionistic functions. To reason about the latter, we have had to develop a step-indexed Kripke logical relation with state transition systems (STS's) that allows us to reason about how an affinely-treated function may have been used.

A translation is said to be fully abstract if two source components cannot be distinguished by any source context if and only if their translations cannot be distinguished by any target context. In recent work, Ahmed and Blume [1] showed how to prove full abstraction of a CPS translation from STLC to System F using purely operational techniques that have the promise to scale well to realistic languages and compilers. The key to their full abstraction result is their CPS type translation which leverages answer-type polymorphism to ensure that a target-level computation (which is assigned a type of the form $\forall\alpha.(\tau \to \alpha) \to \alpha$) can only produce an answer by invoking its continuation. An important feature of their proof technique is the use of a multi-language semantics that formalizes interoperability between the source and target language of the CPS transform. The essence of why their multi-language approach works for CPS is due to an isomorphism between the types τ and $\forall\alpha.(\tau \to \alpha) \to \alpha$—the same isomorphism as presented in Wadler [5]. This isomorphism requires that the parametric condition $k\ (v\ [\tau']\ f) = v\ [\tau_k]\ (k \circ f)$ hold, where $k : \tau' \to \tau_k$, $v : \forall\alpha.(\tau \to \alpha) \to \alpha$, and $f : \tau \to \tau'$.

As Wadler noted, the parametric condition does not hold in the presence of effects such as non-termination. The intuition for the latter is as follows: Let v be a function that calls its argument multiple times, $v = \lambda[\alpha](g : \mathsf{int} \to \alpha).(g\ 0); (g\ 1); (g\ 2)$. If we set f to the identity function, then $v\ [\mathsf{int}]\ id$ will evaluate to the value 2. Now imagine k is a function that diverges when its input is 0 and otherwise behaves like identity, $k = \lambda(x : \mathsf{int}).\mathit{if0}\ x\ \mathit{then}\ \Omega\ \mathit{else}\ x$. On the left side of the parametric condition, k will be applied to 2, evaluating to 2. However, on the right side, v will apply $k \circ id$ to the values $0, 1$, and 2, resulting in divergence from the first application of $k \circ id$.

The failure of the parametric condition in an effectful setting is related to why the use of answer-type polymorphism alone does not yield a fully abstract CPS translation. Informally, a computation of type $\forall\alpha.(\tau \to \alpha) \to \alpha$ may invoke its continuation multiple times which, in the presence of effects, may result in target-level observations that are not possible in the source language (since source computations must use their continuation exactly once). Thus, for fully abstract CPS translation of a language with effects one must use a type translation that can ensure that target-level computations use their continuations linearly [2].

We consider fully abstract CPS translation for a language with recursive types. To scale the Ahmed-Blume technique to a language with effects, we need more than just linearly-used continuations: we also need answer-type polymorphism to achieve interoperability between the source and target of CPS. This time, however, we leverage an isomorphism between the types τ and $\forall\alpha.(\tau \to \alpha) \multimap \alpha$, where the latter makes use of affine functions ($\multimap$) to ensure that the continuation cannot be invoked more than once. Adapting the CPS translation to affinely-treat continuations eliminates the problematic v we used above from the range of the translation, leaving us to formally prove that the parametric condition holds when $v : \forall\alpha.(\tau \to \alpha) \multimap \alpha$. The standard approach for proving such free theorems in an operational manner is to define a step-indexed binary logical relation and use it to show that $k\ (v\ [\tau']\ f)$ is observably equivalent to $v\ [\tau_k]\ (k \circ f)$. This is problematic in the presence of affinity: logical relations are built by assigning semantic meanings to types, but what are the semantics of affinely-treated terms at a given type? At base types, the semantics of affinely-treated terms should be no different than that of their unrestricted counterparts. However, affinely-treated functions seem to require a different semantics than their unrestricted counterparts: the semantics should reflect the fact that the former will not be applied more than once—that is, there exists a unique input value to which an affinely-treated function will be applied. This treatment has precedent in substructural denotational models [4] where, unlike unrestricted functions of the same type, affinely-treated functions need only be defined for one unique input value. The traditional semantic interpretation of an unrestricted function type in a logical relation is hence an over-generalization of an affinely-treated function type because it parameterizes over all possible inputs to the function, whereas affinely-treated functions should only require a definition for one input value.

To make this concrete, consider trying to prove the parametric condition using a logical relation that treats affinely-treated functions of type $\tau \to \tau'$ in the same way as unrestricted functions of that type. Specifically, let us try to prove one direction of the above equivalence, that $k\ (v\ [\tau']\ id) \precsim v\ [\tau_k]\ k : \tau_k$, where for simplicity we've let $f = id$. To do so, we assume the left side terminates and are required to show that the right side does as well. Leveraging the fundamental property of logical relations, which states that a well-typed term is logically related to itself, we get that $v \precsim v : \forall\alpha.(\tau \to \alpha) \multimap \alpha$. This leaves us to show, after eschewing polymorphic-related intricacies, that the left argument to v approximates the right argument, that is, $id \precsim k : \tau \to \alpha$. We hence assume that $id\ v_1 \Downarrow^*$, for an arbitrary v_1 and v_2 such that $v_1 \precsim v_2 : \tau$, and must show that $k\ v_2$ terminates. This is not provable since we only know that $k\ v_a \Downarrow^*$ where $v\ [\tau']\ id \Downarrow^*\ v_a$, which gives us no insight into the behavior of $(k\ v_2)$ when $v_1 \neq v_a$. To remedy this, the semantic interpretation of functions that are affine resources must be able to work with a constrained set of possible input values. Intuitively, this change would allow us to select the set of inputs that must be considered when proving $id \precsim k : \tau \to \alpha$ to be all $v_1 \precsim v_2 : \tau$ such that $v_1 = v_a$.

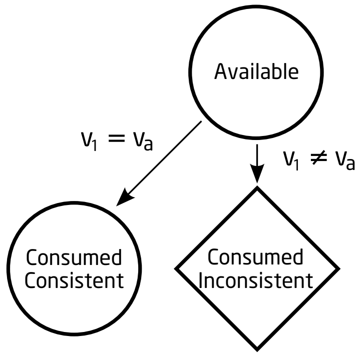

Figure 1: STS for *id* resource

To prove the parametric condition when $v : \forall\alpha.(\tau \to \alpha) \multimap \alpha$, we have developed a step-indexed Kripke logical relation that borrows many ideas from existing Kripke models of mutable references [3]. In our logical relation, applying an affine function (of type $\tau \multimap \tau'$) generates a new resource for which we create a fresh island in the world. Each island contains a state-transition system (STS) that tracks the availability and usage of its affine resource.[1] These STS's allow us to track whether an affinely-treated function is Available or has been Consumed; depending on the input to the function, the STS is moved to a Consumed-Consistent or Consumed-Inconsistent state. The Consistent/Inconsistent states are essentially a device that allows us to do classical reasoning: either id and k were applied to the "good" inputs (i.e., inputs $v_1 \precsim v_2 : \tau$ where $v_1 = v_a$) putting us in a Consistent state in which we can complete the proof; or they were applied to "bad" inputs (i.e., $v_1 \precsim v_2 : \tau$ where $v_1 \neq v_a$) in which case we will be able to prove a contradiction. To prove this contradiction in the case of our free theorem, we must also make use of the ability to pick a relation for α. Thus, the semantics of polymorphism, affine functions, and affinely-treated intuitionistic functions must all be leveraged in the proof of the parametric condition. Figure 1 depicts the STS we use for the *id* resource in our proof of the above example. We note that the creation of new resources in the world does not have an operational counterpart—it is akin to ghost state. Our logical relation uses biorthogonality ($\top\top$) to ensure completeness with respect to contextual equivalence. Using $\top\top$ in the presence of substructural types was subtle and as far as we know, this is the first logical relation for a substructural type system to make use of $\top\top$. This is currently work in progress.

References

[1] A. Ahmed and M. Blume. An equivalence-preserving CPS translation via multi-language semantics. In *International Conference on Functional Programming (ICFP), Tokyo, Japan*, pages 431–444, Sept. 2011.

[2] J. Berdine, P. O'Hearn, U. Reddy, and H. Thielecke. Linear continuation-passing. *Higher Order Symbol. Comput.*, 15(2-3):181–208, 2002.

[3] D. Dreyer, G. Neis, and L. Birkedal. The impact of higher-order state and control effects on local relational reasoning. *Journal of Functional Programming*, 22(4&5):477–528, 2012.

[4] J. Y. Girard, P. Taylor, and Y. Lafont. *Proofs and types.* Cambridge University Press, New York, NY, USA, 1989.

[5] P. Wadler. Theorems for free! In *ACM Symposium on Functional Programming Languages and Computer Architecture (FPCA)*, Sept. 1989.

[1]The STS is uninteresting when the argument of the affine function is of base type; this corresponds to the observation that there should be no difference between affinely-treated terms and unrestricted terms of base type.