Logical Step-indexed Logical Relations

Yiyang Guo

September 2022

Following Harper(2022)[3], we take recursive type as the running example.

0.1 Recap: Step-indexing

Self-referential nature of recursive type poses a difficulty in naive computability method. Harper(2022)[3] explains how step-indexing solves the problem. Compared to the naive computability method, the proof has two extra components:

1. Define a well-founded logical relation by stratifying the logical predicates using step-indices. Specifically, for the recursive type in question, the predicate is given as:

   $$\mathsf{fold}_{X.A}(V) \in^{(n+1)} \mathsf{rec}(x.A) \iff V \in^{(n)} [\mathsf{rec}(X.A)/X]A$$

   Step-index is interpreted as the number of remaining computation steps that we are allowed to observe. Then trivially, all predicates hold at step-index 0 and we can show anti-monotonicity (See Harper(2022)[3]) holds.
   The usual computability predicate is now defined to mean that some property holds for all finite step-indices[1]. Formally:

   $$\Gamma \gg M \in A \iff \forall n \geq 0, \Gamma \gg^{(n)} M \in A$$
   $$\Gamma \gg^{(n)} M \in A \iff \gamma \in^{(n)} \Gamma, \text{ then } \hat{\gamma}(M) \in^{(n)} A$$

2. In the proof of FTLR, where we show the logical predicates are implied from syntactical typing, we induct on the step indices. Since all predicates hold trivially at step-index 0, it suffices to prove the inductive step: assuming the predicates at step-index n, need to show they continue to hold at step-index $n+1$.

0.2 LSLR

Logical step-indexed logical relation(LSLR) by Dreyer et al.[2] follows a similar idea except that it pre-defines a modal logic beforehand, in which we carry out the computability method, hence the name "logical". The added modality nicely hides the proof components related to step-indices.
The benefits of such approach are:

[1] When our goal is to prove something like type safety, this is good enough. See Spies(2021)[4] for the cases when it's not, e.g. termination

1. Encapsulates tedious and error-prone step-index arithmetic into the logic, such that the annoyance is done once and for all. This also works well with the developments of logical frameworks (Iris[1], for example, implements the core idea here).
2. Avoids the asymmetry seen in the direct step-indexed logical relations. This gives rise to simple reasoning that involves binarized logical relations. (TODO: ?how? do a proof)

0.2.1 Step-indexed Model of LSLR

LSLR is interpreted in a Kripke model where worlds are the step-indices. As usual, step-index is the number of remaining computation steps that we are allowed to observe. "Future" worlds correspond to the worlds with smaller step indices.

The basis of the logic is the intuitionistic logic that we usually work with. We get some atomic propositions, standard logical connectives($\wedge\vee\Rightarrow$), first-order quantification($\forall X.P, \exists X.P$), and recursive relations ($\mu r.R$). On top of that, we have a "later" modality for propositions (written as $\triangleright P$, where P is a proposition) [2]. Intuitively $\triangleright P$ says P is true in all strictly future worlds of the current one.
The step-indexed model of the logic is given below:
If $n = 0$:

$$[\![P]\!]n \triangleq \top$$

If $n \geq 1$:

$$\begin{aligned}
[\![P \wedge Q]\!]n &\triangleq [\![P]\!]n \wedge [\![Q]\!]n \\
[\![P \vee Q]\!]n &\triangleq [\![P]\!]n \vee [\![Q]\!]n \\
[\![P \Rightarrow Q]\!]n &\triangleq \forall k \leq n,\, [\![P]\!]k \Rightarrow [\![Q]\!]k \\
[\![\forall X.P]\!]n &\triangleq \forall \gamma \in X, [\![\hat{\gamma}(P)]\!]n \\
[\![\exists X.P]\!]n &\triangleq \exists \gamma \in X, [\![\hat{\gamma}(P)]\!]n \\
[\![\triangleright P]\!]n &\triangleq [\![P]\!]n - 1 \\
[\![\mu r.R]\!]n &\triangleq [\![[\mu r.R/r]R]\!]n
\end{aligned}$$

Claim 1. *Assuming all occurences of r in $\mu r.R$ is under a $\triangleright$-operator, the interpretation is well-founded.*

Proof. Define the "size" of the proposition being interpreted as the number of connectives in it, ignoring the ones under $\triangleright$-operator. The definition is well-founded by a double induction on size and on world. □

Claim 2. *This model enjoys anti-monotonicity, i.e.* $\forall n \geq 0$, $[\![P]\!]n + 1 \Rightarrow [\![P]\!]n$

[2] The original LSLR in Dreyer(2011) [2] also involves constructs pertaining parametricity in System F (second-order quantifiers). Here, we only present the constructs relevant to recursive type.

0.2.2 Inference rules

Under the step-indexed world, we develop a set of inference rules that are sound. The rules pertaining to $\triangleright$-modality are presented below (C is the variable context, the relation variable context, plus the proposition context. See Dreyer(2011)[2] for a more formal account).
Check that the following rules and the standard rules in intuitionistic logic are sound in the model (by induction on worlds).
Notice the LÖB rule corresponds to the induction principle on step-indices that we use in the usual step-index LR proofs $(\llbracket P \rrbracket[n-1] \Rightarrow \llbracket P \rrbracket[n]) \Rightarrow \llbracket P \rrbracket n$.

$$\text{ANTI-MONO}\ \frac{C \vdash P}{C \vdash \triangleright P} \qquad \text{LÖB}\ \frac{C, \triangleright P \vdash P}{C \vdash P} \qquad \triangleright\wedge\ \frac{C \vdash \triangleright(P \wedge Q)}{C \vdash \triangleright P \wedge \triangleright Q} \qquad \triangleright\vee\ \frac{C \vdash \triangleright(P \vee Q)}{C \vdash \triangleright P \vee \triangleright Q} \qquad \triangleright\Rightarrow\ \frac{C \vdash \triangleright(P \Rightarrow Q)}{C \vdash \triangleright P \Rightarrow \triangleright Q}$$

$$\triangleright\forall\ \frac{C \vdash \triangleright(\forall X.P)}{C \vdash \forall X.\, \triangleright P} \qquad \triangleright\exists\ \frac{C \vdash \triangleright(\exists X.P)}{C \vdash \exists X.\, \triangleright P} \qquad \mu\ \frac{C \vdash e \in \mu r.R}{C \vdash e \in [\mu r.R/r]R}$$

As noted by the authors, the set presented in their paper (or here) is not definitive. This is only a canonical set of rules, with which we do not need to reason directly in the model (and therefore with the step-indices) for most proofs. In other words, one may add more sound rules here, if some other proof finds them useful.

0.2.3 Proof

Now that we have LSLR, the proof follows the same 2-step process as the usual step-indexed proof:

1. Define a well-founded logical relation by facilitating $\triangleright$-modality (before: stratify with step-indices)
2. Prove FTLR by Löb induction (before: induction on step-indices).

Here we give the proof of type safety for STLC equipped with recursive type.
The logical relation is given in Figure 1. Note the predicate is well-founded because either the type structure becomes smaller, or the recursive reference to the predicate is put under $\triangleright$-modality. Notice the use of $\triangleright$ corresponds to places where we decrease the "step-index/fuel" in the usual definition of the logical relation.

Theorem 1 (Fundamental Theorem of Logical Relations)**.**
If $\Gamma \vdash M : A$*, then* $\Gamma \gg M \in A$*.*

Proof. By induction on the typing rules.
Variable and unit cases are immediate. Product type introduction and elimination follow easily after expanding the definitions. It relies on the usual intro/elim rule for $\wedge$ connective.
Case-Lam $\dfrac{\Gamma, x : A_1 \vdash M : A_2}{\Gamma \vdash \lambda x.M : A_1 \rightharpoonup A_2}$

$$
\begin{aligned}
\langle\rangle \in_{val} \mathbf{1} &\iff \text{(true)} \\
\langle V_1 \, , V_1 \rangle \in_{val} A_1 \times A_2 &\iff V_1 \in_{val} A_1 \text{ and } V_2 \in_{val} A_2 \\
\lambda x.M \in_{val} A_1 \rightharpoonup A_2 &\iff V_1 \in_{val} A_1 \Rightarrow [V_1/x]M \in_{exp} A_2 \\
\mathsf{fold}_{X.A}(V) \in_{val} \mathsf{rec}(X.A) &\iff \triangleright\, V \in_{val} [\mathsf{rec}(X.A)/X]A \\
M \in_{exp} A &\iff \text{Either } M \mapsto M' \text{ and } \triangleright\, M' \in_{exp} A; \\
&\qquad\quad \text{Or } M \in_{val} A \\
\gamma \in \Gamma &\iff \gamma(x) \in_{val} A_x \text{ for each } \Gamma \vdash x : A_x \\
\Gamma \gg M \in A &\iff \gamma \in \Gamma \Rightarrow \hat{\gamma}(M) \in_{exp} A
\end{aligned}
$$

Figure 1: "Logical" step-indexed logical relation for recursive and simple types

Take $\gamma : \Gamma$. WTS $\lambda x.\hat{M} \in_{exp} \mathsf{rec}(X.A)$.
It suffices to show $\lambda x.\hat{M} \in_{val} \mathsf{rec}(X.A)$, which says $V_1 \in_{val} A_1 \Rightarrow [V_1/x]\hat{M} \in_{exp} A_2$.
By induction, for any $V \in_{val} A_1$, $[V/x]\hat{M} \in_{exp} A_2$.

Case-App $\dfrac{\Gamma \vdash M_1 : A_1 \rightharpoonup A_2 \quad \Gamma \vdash M_2 : A_1}{\Gamma \vdash \mathsf{ap}(M_1, M_2) : A_2}$

Take $\gamma : \Gamma$, by induction $\hat{M_1} \in_{exp} A_1 \rightharpoonup A_2$ and $\hat{M_2} \in_{exp} A_1$. WTS $\mathsf{ap}(\hat{M_1}, \hat{M_2}) \in_{exp} A_2$.
We prove its first-order, general form P:

$$P \triangleq \forall M, \forall N, M \in_{exp} A_1 \rightharpoonup A_2 \wedge N \in_{exp} A_1 \Rightarrow \mathsf{ap}(M, N) \in_{exp} A_2$$

By Löb induction, it suffices to show $\triangleright P \vdash P$.
So assume $\triangleright(\forall M, \forall N, M \in_{exp} A_1 \rightharpoonup A_2 \wedge N \in_{exp} A_1 \Rightarrow \mathsf{ap}(M, N) \in_{exp} A_2)$.
By rule $\triangleright \Rightarrow$, $\triangleright\forall$ and $\triangleright\wedge$, we can expand the assumption to:
$\forall M, \forall N, \triangleright M \in_{exp} A_1 \rightharpoonup A_2 \wedge \triangleright N \in_{exp} A_1 \Rightarrow \triangleright\mathsf{ap}(M, N) \in_{exp} A_2$.
Now WTS P. Take N, M such that $M \in_{exp} A_1 \rightharpoonup A_2 \wedge N \in_{exp} A_1$, WTS $\mathsf{ap}(M, N) \in_{exp} A_2$.
By definition, two cases:

1. $M \in_{val} A_1 \rightharpoonup A_2 \wedge N \in_{val} A_1$. Then M must take the form $\lambda x.M'$ and $[V_1/x]M' \in_{exp}$ for any $V_1 \in_{val} A_2$.
   Then $\mathsf{ap}(M, N) \mapsto [N/x]M' \in_{exp} A_2$. By anti-monotonicity, $\triangleright[N/x]M' \in_{exp} A_2$. By definition $\mathsf{ap}(M, N) \in_{exp} A_2$
2. $M \mapsto M'$(or $N \mapsto N'$) and $\triangleright M' \in_{exp} A_1 \rightharpoonup A_2$ (or $\triangleright N' \in_{exp} A_1$).
   By anti-monotonicity, $\triangleright N \in_{exp} A_1$ (or $\triangleright M \in_{exp} A_1 \rightharpoonup A_2$). Applying the assumption by instantiating with M', N (or M, N'), we have $\triangleright\mathsf{ap}(M', N)$ (or $\triangleright\mathsf{ap}(M, N')$).
   Since $\mathsf{ap}(M, N) \mapsto \mathsf{ap}(M', N)$ (or $\mathsf{ap}(M, N')$), $\mathsf{ap}(M, N) \in_{exp} A_2$.

Case-Fold $\dfrac{\Gamma \vdash M : [\mathsf{rec}(X.A)/X]A}{\Gamma \vdash \mathsf{fold}_{X.A}(M) : \mathsf{rec}(X.A)}$

Take $\gamma \in \Gamma$. By induction, $\hat{M} \in_{exp} [\mathsf{rec}(X.A)/X]A$. WTS $\mathsf{fold}_{X.A}(\hat{M}) \in_{exp} \mathsf{rec}(X.A)$.
Again, we prove its first-order, general form P:

$$P \triangleq \forall N, N \in_{exp} [\mathsf{rec}(X.A)/X]A \Rightarrow \mathsf{fold}_{X.A}(N) \in_{exp} \mathsf{rec}(X.A)$$

By Löb induction, it suffices to show $\triangleright P \vdash P$.
So assume $\triangleright(\forall N, N \in_{exp} [\mathsf{rec}(X.A)/X]A \Rightarrow \mathsf{fold}_{X.A}(N) \in_{exp} \mathsf{rec}(X.A))$.
By rule $\triangleright \Rightarrow$ and $\triangleright\forall$, we have $\forall N, \triangleright N \in_{exp} [\mathsf{rec}(X.A)/X]A \Rightarrow \triangleright\mathsf{fold}_{X.A}(N) \in_{exp} \mathsf{rec}(X.A)$.
WTS P. Take $N \in_{exp} [\mathsf{rec}(X.A)/X]A$. WTS $\mathsf{fold}_{X.A}(N) \in_{exp} \mathsf{rec}(X.A)$
By definition, two cases:

1. Either $N \mapsto N'$ and $\triangleright N' \in_{exp} [\mathsf{rec}(X.A)/X]A$. Applying the assumption by instantiating with N', we have $\triangleright\mathsf{fold}_{X.A}(N') \in_{exp} \mathsf{rec}(X.A)$.
   Since $\mathsf{fold}_{X.A}(N) \mapsto^1 \mathsf{fold}_{X.A}(N')$, $\mathsf{fold}_{X.A}(N) \in_{exp} \mathsf{rec}(X.A)$.
2. Or $N \in_{val} [\mathsf{rec}(X.A)/X]A$. By anti-monotnicity $\triangleright N \in_{val} [\mathsf{rec}(X.A)/X]A$. Then by definition $\mathsf{fold}_{X.A}(N) \in_{val} \mathsf{rec}(X.A)$, which implies $\mathsf{fold}_{X.A}(N) \in_{exp} \mathsf{rec}(X.A)$.

Case-Unfold $\dfrac{\Gamma \vdash M : \mathsf{rec}(X.A)}{\Gamma \vdash \mathsf{unfold}(M) : [\mathsf{rec}(X.A)/X]A}$

Take $\gamma \in \Gamma$. WTS $\mathsf{unfold}(\hat{M}) \in_{exp} [\mathsf{rec}(X.A)/X]A$.
Again, we prove its first-order, general form P:

$$P \triangleq \forall N, N \in_{exp} \mathsf{rec}(X.A) \Rightarrow \mathsf{unfold}(N) \in_{exp} [\mathsf{rec}(X.A/X)]A$$

By Löb induction, it suffices to show $\triangleright P \vdash P$.
Assume $\triangleright(\forall N, N \in_{exp} \mathsf{rec}(X.A) \Rightarrow \mathsf{unfold}(N) \in_{exp} [\mathsf{rec}(X.A/X)]A)$.
By rule $\triangleright \Rightarrow$ and $\triangleright\forall$, we have $\forall N, \triangleright N \in_{exp} \mathsf{rec}(X.A) \Rightarrow \triangleright\mathsf{unfold}(N) \in_{exp} [\mathsf{rec}(X.A/X)]A$.
WTS P. Take $N \in_{exp} \mathsf{rec}(X.A)$, WTS $\mathsf{unfold}(N) \in_{exp} [\mathsf{rec}(X.A/X)]A$.
By definition, two cases:

1. Either $N \mapsto N'$ and $\triangleright N' \in_{exp} \mathsf{rec}(X.A)$. Apply the assumption by instantiating with N', we get $\triangleright\mathsf{unfold}(N') \in_{exp} [\mathsf{rec}(X.A/X)]A$.
   Since $\mathsf{unfold}(N) \mapsto \mathsf{unfold}(N')$, by definition $\mathsf{unfold}(N) \in_{exp} [\mathsf{rec}(X.A/X)]A$.
2. Or $N \in_{val} \mathsf{rec}(X.A)$. Then N must take the form $\mathsf{fold}_{X.A}(V)$ and $\triangleright V \in_{val} [\mathsf{rec}(X.A/X)]A$. Then $\mathsf{unfold}(N) = \mathsf{unfold}(\mathsf{fold}_{X.A}(V)) \mapsto V$. By definition, $\mathsf{unfold}(N) \in_{exp} [\mathsf{rec}(X.A/X)]A$

□

References

[1] Ralf Jung 0002, Robbert Krebbers, Jacques-Henri Jourdan, Ales Bizjak, Lars Birkedal, and Derek Dreyer. Iris from the ground up: A modular foundation for higher-order concurrent separation logic. *Journal of Functional Programming*, 28, 2018.

[2] Derek Dreyer, Amal Ahmed, and Lars Birkedal. Logical step-indexed logical relations. *Computing Research Repository - CORR*, 7, 03 2011.

[3] Robert Harper. Step-indexed logical relations. unpublished lecture note. url `https://www.cs.cmu.edu/~rwh/courses/atpl/pdfs/step-idx.pdf`. Spring 2022.

[4] Simon Spies, Neel Krishnaswami, and Derek Dreyer. Transfinite step-indexing for termination. *Proc. ACM Program. Lang.*, 5(POPL), jan 2021.