Shuntaro Katsuda

Undergraduate School of Informatics, Kyoto University

Saturday 23rd November, 2019

4 D F 4 D F 4 D F 9 Q C

Staged Dependent Lambda Calculus with Equality Types

y of terms, types and kinds

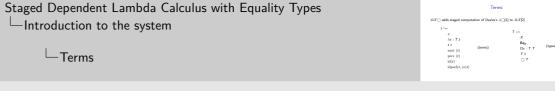
raluation of terms

eferences

Terms

 λLF adds staged computation of Davies's λ [1] to λLF [2]

$$t :=$$
 x
 $\lambda x : T.t$
 $t t$
 $\operatorname{next}(t)$
 $\operatorname{prev}(t)$
 $\operatorname{id}(t)$
 $\operatorname{idpeel}(t,(x)t)$
 $T :=$
 X
 Eq_{T}
 $\operatorname{T} t$
 $T t$
 $T t$
 $T t$



hogehoge

Equality of terms, types and kinds

Do not allow equivalence of terms via code embedding inside next.

$$\frac{\Gamma \vdash^{(n)} t : T \qquad \Gamma \vdash^{(n)} T :: * \qquad m \le 1}{\Gamma \vdash^{(n)} \text{prev (next } (t)) \equiv_m t : T}$$
 (Q-\cap Beta)

Disallow

$$\Gamma \vdash^{(n)} \text{next (next (3 + prev (next (5))))} \equiv_0 \text{next (next (3 + 5))}$$

Allow

$$\Gamma \vdash^{(n)} \text{next } (3 + \text{prev } (\text{next } (5))) \equiv_0 \text{next } (3 + 5)$$

4 D > 4 A > 4 E > 4 E > E = 90 A

Staged Dependent Lambda Calculus with Equality Types

Equality of terms, types and kinds

Do not allow equivalence of turns via code embedding stable and.

The proof of terms, types and kinds

Equality of terms concerning stages

Equality of terms concerning stages

The \(\cap \)Beta rule works well in disallowing code embedding inside next.

Disallow

 $\Gamma \vdash^{(n)} \text{next (next (3 + prev (next (5))))} \equiv_0 \text{next (next (3 + 5))}$

$$\Gamma \vdash^{(n)} TODO$$

Staged Dependent Lambda Calculus with Equality Types Equality of terms, types and kinds

The \bigcirc BETA rule works well in disallowing code embedding inside next. Disallow $\Gamma \vdash^{(6)} \text{next (next (3+prev (next (5))))} \equiv_0 \text{next (next (3+5))}$ $\Gamma \vdash^{(6)} TODO$ Because of dependent types, some types include terms. Stages need to be considered when comparing types.

Example

$$\Gamma \vdash^{(n)} \text{Vector } 3 \equiv_0^? \text{Vector } (1+2)$$

Staged Dependent Lambda Calculus with Equality Types —Equality of terms, types and kinds

Equivalence of types concerning stages



前回は, type と kind について equivalence の index がありませんでしたが, よく考えたら必要なことが判明しました. これをどう定義したかについてお話します. とりあえずこの例では同じにしたいです.

Equality of terms, types and kinds

Trickier Example

Example

$$\Gamma \vdash^{(n)} \text{next (Vector 3)} \equiv_0^? \text{Vector } (1+2)$$

policy

Compare terms inside types at the current equivalence depth.

Staged Dependent Lambda Calculus with Equality Types -Equality of terms, types and kinds

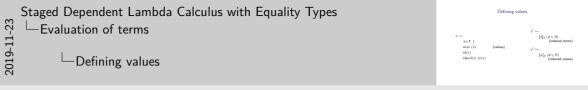
 $\Gamma \vdash^{(4)} \text{next (Vector 3)} \equiv_0^7 \text{Vector } (1+2)$

Trickier Example

Vector 3 が next の中に入っています.

Trickier Example

Defining values



Equality of terms, types and kinds

Evaluation of terms

Meaning of the index

$$\frac{\llbracket t \rrbracket_{d+1} \to \llbracket t' \rrbracket_{d+1} \qquad d \le 1}{\llbracket \operatorname{next} (t) \rrbracket_d \to \llbracket \operatorname{next} (t') \rrbracket_d}$$

$$\frac{\llbracket t \rrbracket_{d+1} \to \llbracket t' \rrbracket_{d+1} \qquad d \le 1}{\llbracket \operatorname{prev}(t) \rrbracket_d \to \llbracket \operatorname{prev}(t') \rrbracket_d}$$



−Evaluation of terms

└─Meaning of the index

Staged Dependent Lambda Calculus with Equality Types

Meaning of the index

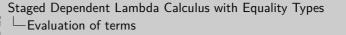
$$\begin{split} & \underbrace{[t]_{d+1} \rightarrow [t']_{d+1}}_{\text{[next }(t)]_d} \xrightarrow{d \leq 1} \\ & \underbrace{[\text{next }(t')]_d}_{\text{[next }(t')]_d} \\ & \underbrace{[t]_{d+1} \rightarrow [t']_{d+1}}_{\text{[prev }(t)]_d} \xrightarrow{d \leq 1} \\ & \underbrace{[\text{prev }(t)]_d}_{\text{[prev }(t')]_d} \end{split}$$

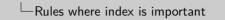
(t')]_d

000000

$$\frac{\llbracket t \rrbracket_d \to \llbracket t' \rrbracket_d \qquad d \le 1}{\llbracket \operatorname{prev} \ (\operatorname{next} \ (t)) \rrbracket_d \to \llbracket t' \rrbracket_d}$$

$$\frac{[\![t]\!]_d \to [\![t']\!]_d \qquad d \le 1}{[\![\operatorname{next (prev }(t))]\!]_d \to [\![t']\!]_d}$$







Rules where index is important

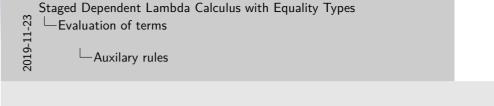




Evaluation of terms

Equality of terms, types and kinds

Introduction to the system



Auxilary rules

References

TODO

References

Rowan Davies and Frank Pfenning. "A modal analysis of staged computation"

References



Rowan Davies and Frank Pfenning. "A modal analysis of staged computation". In: Conference Record of the Annual ACM Symposium on Principles of Programming Languages. Vol. Conference. 3. May 1996, pp. 258–270. DOI: 10.1145/382780.382785. URL: http://portal.acm.org/citation.cfm?doid=382780.382785.



Benjamin C Pierce. Advanced Topics in Types and Programming Languages. Cambridge, Mass.: The Mit Press, 2019. DOI: 10.7551/mitpress/1104.001.0001.

4 D > 4 D > 4 D > 4 D > 5 P 9 Q P