Monotonic Gradual Typing in a Common Calculus

Ben Trovato*
Institute for Clarity in Documentation
Dublin, Ohio
trovato@corporation.com

John Smith
The Thørväld Group
jsmith@affiliation.org

ABSTRACT

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings.[1]

CCS CONCEPTS

• **Computer systems organization → Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

KEYWORDS

ACM proceedings, LaTeX, text tagging

ACM Reference Format:
Ben Trovato and John Smith. 1997. Monotonic Gradual Typing in a Common Calculus. In *Proceedings of ACM Woodstock conference, El Paso, Texas USA, July 1997 (WOODSTOCK'97)*, ?? pages.
https://doi.org/10.475/123_4

1 COMPLETE TRANSLATION FOR MONOTONIC

1.1 Monotonic cast static and dynamic rules

$$\frac{\Gamma\ \sigma\ K \vdash e : t'}{\Gamma\ \sigma\ K \vdash \triangleleft t \triangleright\ e : t}\ {}^{W10}$$

$$K\ \triangleleft t \triangleright\ a\ \sigma \longrightarrow K'\ a'\ \sigma'\ \ \mathrm{moncast}(a, t, \sigma, K) = K'\ a'\sigma'\ {}^{E11}$$

$$E ::= \ldots \mid \triangleleft t \triangleright\ E$$

1.2 Monotonic translation for bidirectional expressions

$$M[\![x]\!]_\Gamma = x$$

$$M[\![e_1.m(e_2)]\!]_\Gamma = e_1'@m_{\star\to\star}(e_2') \quad (\textbf{if}\ K, \Gamma \vdash e : \star \wedge M[\![e_1]\!]_\Gamma = e_1' \wedge M(\!|e_2|\!)^{\star}_{\Gamma} = e_2')$$

$$M[\![e_1.m(e_2)]\!]_\Gamma = e_1'.m_{D_1\to D_2}(e_2') \quad (\textbf{if}\ K, \Gamma \vdash e : C \wedge m(D_1) : D_2 \in K(C) \wedge M[\![e_1]\!]_\Gamma = e_1' \wedge M(\!|e_2|\!)^{D_1}_{\Gamma} = e_2')$$

$$M[\![\textbf{new}\ C(e_1 \ldots)]\!]_\Gamma = \textbf{new}\ C(e_1' \ldots) \quad (\textbf{if}\ f_1 : t_1 \in K(C) \wedge e_1' = M(\!|e_1|\!)^{t_1}_{\Gamma} \ldots)$$

$$M(\!|e|\!)^{t}_{\Gamma} = e' \quad (\textbf{if}\ K, \Gamma \vdash e : t' \wedge K \vdash t' <: t)$$

$$M(\!|e|\!)^{t}_{\Gamma} = \triangleleft t \triangleright\ e' \quad (\textbf{if}\ K, \Gamma \vdash e : t' \wedge K \vdash t' \not<: t)$$

2 GENERATIVE MONOTONE CASTS

$$\frac{\sigma(a) = C\{a' \ldots\} \qquad \mathrm{tmeet}(a, C, t, \sigma)K = C'\ K' \qquad \sigma' = \sigma[a \mapsto C'\{a' \ldots\}]}{\mathrm{moncast}(a, t, \sigma, K) = K'\ \sigma'}\ {}^{CM}$$

Monotonic cast semantics

$\boxed{\mathrm{tmeet}(t, t', P, K) = t''\ K'}$

$$P ::= \cdot \mid P\ (C, D) \mapsto C'$$

$$\frac{}{\mathrm{tmeet}(C, \star, P, K) = C\ K}\ {}^{TM1} \qquad \frac{}{\mathrm{tmeet}(\star, C, P, K) = C\ K}\ {}^{TM2}$$

$$\frac{K'\ a'\sigma'}{\mathrm{tmeet}(t, t, P, K) = t\ K}\ {}^{TM3}$$

$$\frac{\begin{array}{c} C'\ \text{fresh} \\ (C, D) \notin P \qquad P' = P\ (C, D) \mapsto C' \qquad \mathrm{mtype}(C, K, =)\mathrm{mt}.. \\ \mathrm{mtype}(D, K, =)\mathrm{mt}'.. \qquad \mathrm{mmeet}(\mathrm{mt}.., \mathrm{mt}'.., P', K) = \mathrm{mt}''.. K' \\ K'' = K'\ \mathrm{typegen}(\mathrm{mt}''.., C') \end{array}}{\mathrm{tmeet}(C, D, P, K) = C'\ K''}\ {}^{TM4}$$

$$\frac{P(C, D) = C'}{\mathrm{tmeet}(C, D, P, K) = C'\ K}\ {}^{TM5}$$

The `tmeet` function

2.1 Meet function

The `mmeet` function is used by the `tmeet` functions to perform the meet over the typing of each method within a class definition. The `mmeet` function also takes four arguments, the method signatures of the original class mt.., the method signatures of the cast class mt′.., the environment P, a class table K, and outputs method types

*Dr. Trovato insisted his name be first.
[1]This is an abstract footnote

WOODSTOCK'97, July 1997, El Paso, Texas USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

$mt''..$ and a class table K'.

$$\frac{}{\mathsf{mmeet}(mt..,\cdot,\Gamma,K) = mt..\ K}\ \text{MM1} \qquad \frac{}{\mathsf{mmeet}(\cdot,mt..,\Gamma,K) = mt..\ K}\ \text{MM2}$$

$$\frac{\begin{array}{c} m(t_3) : t_4 \in mt_2 \\ \mathsf{tmeet}(t_3,t_1,\Gamma,K) = t_5\ K' \qquad \mathsf{tmeet}(t_2,t_4,\Gamma,K') = t_6\ K'' \\ \mathsf{mmeet}(mt_1..,mt_2..,\Gamma,K'') = mt_3..\ K''' \end{array}}{\mathsf{mmeet}(m(t_1) : t_2\ mt_1..,mt_2..,\Gamma,K) = n(t_5) : t_6\ mt_3..\ K'''}\ \text{MM6}$$

2.2 Monotonic class generation

The monWrap function generates the wrapper for the monotonic translations, when a monotonic cast is required.

$\mathsf{monWrap}(C, md.., mt.., mt'.., D, K) =$
class D {
$f : t \qquad \forall\, f : t \in \mathsf{fields}(K, C)\ .$
$m(x : \star) : \star\ \{ \lhd \star \rhd \mathsf{this}.m_{t_1 \to t_2}(\lhd t_1 \rhd x)\ \}\ \forall\, m\ .\ m(x : \star) : \star\ \{e\} \in md.. \wedge\ m(C_1) : C_2 \in mt'..$
$m(x : C_1) : C_2\ \{\lhd C_2 \rhd [(\lhd \star \rhd x)/x]e'\ \} \qquad \forall\, m\ .\ m(x : \star) : \star\ \{e\} \in md.. \wedge\ m(C_1) : C_2 \in mt'..$
}

2.3 Monotonic equivalent type generation

The typegen function is used by the tmeet function to generate the new classes of the type produced by the meet operation.

$\mathsf{typegen}(mt.., D) =$
class D {
$m(x : t) : t'\ \{\lhd t' \rhd x\}\ \forall\, m\ .\ m(t) : t' \in mt..$
}