Sized Type Theory

by Sven Nilsen, 2020

In this paper I introduce a type theory for sized types, which permits inference depending on type size.

Symbols	Type	Description	Language
A, B, C	-	Grammar expression	Parsing
$A \mid B$	-	Selection grammar rule	Parsing
()[]{}	-	Parentheses	Parsing
x, y, z, i, j, k, f, g	-	Variable expression	Computation
f(x)	-	Application	Computation
bool	type	Boolean	Boolean algebra
¬, ∧, ∨, ⊻, ==, =>	bool → bool, (bool, bool) → bool	NOT, AND, OR, XOR, EQ, IMPL	Boolean algebra
T, U, V	type	Generic type	Type theory
if A { B } else { C }	bool → T → T	If (with optional else-ifs)	Logic
∀ A { B }	bool	For-all	Logic
∃ A { B }	bool	There-exists	Logic
A + B	(nat, nat) → nat	Addition	Arithmetic
A · B	(nat, nat) → nat	Multiplication	Arithmetic
<=>	-	Substitutional equivalence	Meta-language
α	-	Quantifier symbol	Meta-language
⊥	-	Bottom type	Type theory
type	$type_1$	Type universe zero	Type theory
$type_n$	$type_{n+1}$	Type universe	Type theory
()	type	Unit type	Type theory
A : B	$(type_n, type_{n+1}) \rightarrow bool$	Judgement	Type theory
A + B, (A, B), A → B	(type, type) → type	Sum, product, exponential type	Type theory
A := B	-	Definitional equality	Type theory
$\|A\|$	type → nat	Type size operator	Sized type theory
A[B]	(type, type) → type	Index operator	Sized type theory
A[1..]	type → type	Tail type operator	Sized type theory
A → → B	(type, type) → type	Isomorphism operator	Sized type theory
A ~:= B	type → → type	Definitional space	Sized type theory
A ~= B	(type, type) → type	Equivalence	Sized type theory
A ~ 0, A ~ 1	(type ~= type) → type	Path equivalence endpoints	Sized type theory
A^{-1}	(type → → type) → (type → → type)	Isomorphism inverse	Sized type theory

Standard path semantical notation is permitted. Typing rules for path semantics is not covered in here. This includes lambda calculus, arbitrary subtypes and Higher Order Operator Overloading.

Quantifier symbol

α	$\Leftrightarrow$	$\forall \mid \exists$

Grammar substitutions

$\alpha\ A, B\ \{ C \}$	$\Leftrightarrow$	$\alpha\ A\ \{ \alpha\ B\ \{ C \} \}$	Nested loop
$\alpha\ A\ \{ B[A] \}$	$\Leftrightarrow$	$\alpha\ A\ \lvert B \rvert\ \{ B[A] \}$	Inferred loop
$\alpha\ A : B\ \{ C \}$	$\Leftrightarrow$	$\alpha\ A\ \{ (A : B) => C \}$	Subtype loop
$A + B + C$	$\Leftrightarrow$	$A + (B + C)$	Addition
$A \cdot B \cdot C$	$\Leftrightarrow$	$A \cdot (B \cdot C)$	Multiplication
$A + B + C$	$\Leftrightarrow$	$A + (B + C)$	Sum type
(A, B, C)	$\Leftrightarrow$	$(A, (B, C))$	Product type

Type size axioms

$\lvert \bot \rvert == 0$	Bottom type
$\forall x : bool\ \{ \lvert x \rvert == 1 \}$	Booleans
$\forall x : nat\ \{ \lvert x \rvert == 1 \}$	Natural numbers
$\forall x, y\ \{ \lvert x + y \rvert == \lvert x \rvert + \lvert y \rvert \}$	Sum type
$\forall x, y\ \{ \lvert (x, y) \rvert == \lvert x \rvert \cdot \lvert y \rvert \}$	Product type
$\forall x, y\ \{ \lvert x \to y \rvert == \lvert y \rvert^{\lvert x \rvert} \}$	Function/exponential type
$\forall x, y\ \{ \text{if } \lvert x \rvert == \lvert y \rvert\ \{ \lvert x \to\to y \rvert == \lvert x \rvert! \} \text{ else } \{ 0 \} \}$	Isomorphism type size
$\forall x, y\ \{ (x \sim:= y) => (\lvert x \rvert == \lvert y \rvert) \}$	Space type size

Type judgements

$\forall x\ \{ (\lvert x \rvert == 1) == \exists y\ \{ (x : y) \wedge (y : type) \} \}$	Primitive type
$\forall x, y, i\ \{ x[i] : (x + y) \}$	Sum type
$\forall x, y, i, j\ \{ (x[i], y[j]) : (x, y) \}$	Product type
$\forall x, y, i, f : x \to y\ \{ f(x[i]) : y \}$	Function/exponential type
$\forall x, y\ \{ (x \sim:= y) : (x : nat \wedge (< \lvert x \rvert) \to\to x) \}$	Space type
$\forall x, y, i, j\ \{ (x[i] \sim= y[j]) : (x \sim= y) \}$	Equivalence type
$\forall x, y\ \{ \forall i\ \{ (x[i] == y) == (y : x) \} \}$	Isomorphism
$\forall f : x \to y, g : x \to\to z\ \{ f[g \to id] : z \to y \}$	Asymmetric path transport
$\forall g : x \to\to y\ \{ g^{-1} : y \to\to x \}$	Isomorphism inverse
$\forall x, y\ \{ ((x \sim= y) \sim 0) : x \}$	Path start point
$\forall x, y\ \{ ((x \sim= y) \sim 1) : y \}$	Path end point

Space orders

$\forall x, y, z\ \{ (x \sim:= y + z) \wedge (\lvert y \rvert == 1) => (x[0] == y) \wedge (x[1..] == z) \}$	Sum type
$\forall x, y, z\ \{ (x \sim:= (y, z)) => \forall i, j\ \{ x[i + j \cdot \lvert y \rvert] == (y[i], z[j]) \} \}$	Product type

Equivalence operator overloading

$\forall x : X, y : Y, f : X \to Y\ \{ f(x \sim= y) == (f(x) \sim= f(y)) \}$	Function application
$\forall x : X, y : Y, f : X \to\to Y\ \{ f[x \sim= y] == (f[x] \sim= f[y]) \}$	Space indexing

Equivalence construction

$\forall x\ \{ x \sim= x \}$	By reflection
$\forall f : X \to Y, g : X \to\to Z\ \{ (f \sim= f[g \to id]) : type \}$	By asymmetric transport
$\forall g : X \to\to Y\ \{ (g \sim= g^{-1}) : (Y \sim= X) \to\to (X \sim= Y) \}$	By isomorphism inverse

Booleans

$bool \sim:= false + true$

Natural numbers

$nat \sim:= 0 + 1 + 2 + \dots$