

CS 671 Automated Reasoning

Quotient Types and Inductive Types



QUOTIENT TYPES: USER-DEFINED EQUALITY

• Representation of equivalence classes

- Members of $x, y : T // E$ are elements of T (but $x, y : T // E \not\subseteq T$)
- Equality $s=t$ redefined as $E[s, t/x, y]$
- E must be type of an equivalence relation

Syntax:

Canonical: $x, y : T // E$

Noncanonical: —

Semantics:

- $x_1, y_1 : T_1 // E_1 = x_2, y_2 : T_2 // E_2$ if $T_1 = T_2$ and there are terms p_1, p_2, r, s, t and variables x, y, z , which occur neither in E_1 nor in E_2 such that

$$p_1 \text{ in } \forall x : T_1. \forall y : T_1. E_1[x, y/x_1, y_1] \Rightarrow E_2[x, y/x_2, y_2],$$

$$p_2 \text{ in } \forall x : T_1. \forall y : T_1. E_2[x, y/x_2, y_2] \Rightarrow E_1[x, y/x_1, y_1],$$

$$r \text{ in } \forall x : T_1. E_1[x, x/x_1, y_1],$$

$$s \text{ in } \forall x : T_1. \forall y : T_1. E_1[x, y/x_1, y_1] \Rightarrow E_1[y, x/x_1, y_1],$$

$$\text{and } t \text{ in } \forall x : T_1. \forall y : T_1. \forall z : T_1. E_1[x, y/x_1, y_1] \Rightarrow E_1[y, z/x_1, y_1] \Rightarrow E_1[x, z/x_1, y_1]$$

- $s = t$ in $x, y : T // E$ if $x, y : T // E$ type, s in T , t in T ,
and there is some term p in $E[s, t/x, y]$

QUOTIENT TYPES: PROOF THEORY

Proof rules must manage implicit information

- We “know” $E[s, t/x, y]$ if $s = t$ in $x, y : T // E$
- Proof term for $E[s, t/x, y]$ can only be used non-computationally
- **Hidden assumptions** generated by decomposing equalities in hypotheses

$$\Gamma, v: s = t \in x, y : T // E, \Delta \vdash C \text{ [ext } u]$$

by `quotient_equalityElimination` $i \ j \ v'$

$$\Gamma, v: s = t \in x, y : T // E, \llbracket v' \rrbracket : E[s, t/x, y], \Delta \vdash C \text{ [ext } u]$$
$$\Gamma, v: s = t \in x, y : T // E, \Delta \vdash E[s, t/x, y] \in \mathbb{U}_j \text{ [Ax]}$$

User-predicates may require **type-squashing**

- $\downarrow P \equiv \{x: \text{Top} \mid P\}$: reduce P to its truth content
- Necessary if there is too much structure on $x, y : T // E$

See Appendix A.3.14 for further details

INDUCTIVE TYPES: RECURSIVE DEFINITION

● Representation of recursively defined datatypes

- Recursive type definition through the equation $X = T[X]$
- Canonical elements are determined by unrolling $T[X]$
- Noncanonical form allows inductive evaluation of elements

● Recursive definition must be well-founded

- Least fixed point semantics
- $T[X]$ must contain a “base” case
- X must only occur positively in $T[X]$

● Extensions possible

- Parameterized, simultaneous recursion
 `rectype $X_1(x_1) = T[X_1]$ and ... $X_n(x_n) = T[X_n]$ select $X_i(a_i)$`
- Co-inductive type `inftype $X = T_X$` : greatest fixed point semantics
- Partial recursive functions `$S \not\rightarrow T$` : unrestricted recursive induction

INDUCTIVE TYPES, FORMALLY

Syntax:

Canonical: $\text{rectype } X = T_X$

Noncanonical: $\text{let}^* f(x) = t \text{ in } f(e)$

Evaluation:

$$\frac{e \downarrow \text{canonical} \quad t[\lambda y. \text{let}^* f(x) = t \text{ in } f(y), e / f, x] \downarrow \text{val}}{\text{let}^* f(x) = t \text{ in } f(e) \downarrow \text{val}}$$

Termination of $\text{let}^ f(x) = t \text{ in } f(e)$ requires e in $\text{rectype } X = T[X]$*

Semantics:

- $\text{rectype } X_1 = T_{X_1} = \text{rectype } X_2 = T_{X_2}$
if $T_{X_1}[X/X_1] = T_{X_2}[X/X_2]$ for all types X
- $s = t$ in $\text{rectype } X = T_X$ if $\text{rectype } X = T_X$ type and
 $s = t$ in $T_X[\text{rectype } X = T_X/X]$

See Appendix A.3.11 for further details

SUMMARY: STANDARD NUPRL TYPES

Function Space	$S \rightarrow T, x : S \rightarrow T$	$\lambda x . t, f t$
Product Space	$S \times T, x : S \times T$	$\langle s, t \rangle, \text{let } \langle x, y \rangle = e \text{ in } u$
Disjoint Union	$S + T$	$\text{inl}(s), \text{inr}(t), \text{case } e \text{ of } \text{inl}(x) \mapsto u \mid \text{inr}(y) \mapsto v$
Universes	\mathbb{U}_j	— <i>types of level j</i> —
Equality	$s = t \in T$	Ax
Empty Type	Void	$\text{any}(x),$ — <i>no members</i> —
Atoms	Atom	“ <i>token</i> ”, if $a=b$ then s else t
Numbers	\mathbb{Z}	$0, 1, -1, 2, -2, \dots$ $s+t, s-t, s*t, s \div t, s \text{ rem } t,$ if $a=b$ then s else $t,$ if $i < j$ then s else t $\text{ind}(u; x, f_x . s; \text{base}; y, f_y . t)$
Lists	$i < j$ $S \text{ list}$	Ax $[], t :: \text{list}, \text{list_ind}(L; \text{base}; x, l, f_l . t)$
Inductive Types	rectype $X = T[X]$	$\text{let}^* f(x) = t \text{ in } f(e),$ — <i>members defined by $T[X]$</i> —
Subset	$\{x : S \mid P[x]\},$	— <i>some members of S</i> —
Intersection	$\cap x : S . T[x],$ $x : S \cap T[x]$	— <i>members that occur in all $T[x]$</i> — — <i>members x that occur S and $T[x]$</i> —
Union	$\cup x : S . T[x]$	— <i>members that occur in some $T[x]$, tricky equality</i> —
Quotient	$x, y : S // E[x, y]$	— <i>members of S, new equality</i> —
Very Dep. Functions	$\{f \mid x : S \rightarrow T[f, x]\}$	
Squiggle Equality	$s \sim t$	— <i>a “simpler” equality</i>

IMPORTANT DEFINED TYPES

- Integer ranges: $\mathbb{N} \equiv \{i:\mathbb{Z} \mid 0 \leq i\}$, $\{j \dots\} \equiv \{i:\mathbb{Z} \mid j \leq i\}$,
 $\mathbb{N}^+ \equiv \{i:\mathbb{Z} \mid 0 < i\}$, $\{\dots j\} \equiv \{i:\mathbb{Z} \mid i \leq j\}$
- Logic: $\forall \exists \wedge \vee \Rightarrow \neg$ **True False** (Curry-Howard isomorphism)
- Singleton type: **Unit** $\equiv 0 \in \mathbb{Z}$
- Boolean: $\mathbb{B} \equiv \text{Unit} + \text{Unit}$, $\uparrow b \equiv \text{if } b \text{ then True else False}$
- Top type: **Top** $\equiv \bigcap x:\text{Void}.\text{Void}$
- Subtyping: $S \sqsubseteq T \equiv \forall x:S. x \in T$
- Type Squashing: $\downarrow P \equiv \{\text{True} \mid P\}$
- Recursive functions: **Y** $\equiv \lambda f. (\lambda x.f (x x)) (\lambda x.f (x x))$

See the standard theories of NUPRL 5 for further details