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 $x, y : T//E \not\sqsubseteq T$)
  - 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 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 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 in $\forall x{:}T_1.\ E_1[x, x/x_1, y_1]$,
s in $\forall x{:}T_1.\forall y{:}T_1.\ E_1[x, y/x_1, y_1] \Rightarrow E_1[y, x/x_1, y_1]$,
and t 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

$\Gamma,\, v{:}\, s = t \;\in\; x\,,y : T//E\,,\, \Delta \vdash C$ ⌊ext u⌋
by `quotient_equalityElimination` i j v'
$\Gamma,\, v{:}\, s = t \;\in\; x\,,y : T//E\,,\, [\![v']\!]{:}E[s,t/x,y],\, \Delta \vdash C$ ⌊ext u⌋
$\Gamma,\, v{:}\, s = t \;\in\; x\,,y : T//E\,,\, \Delta,\, x'{:}T,\, y'{:}T \vdash E[x',y'/x,y] \;\in\; \mathbb{U}_j$ ⌊Ax⌋

User-predicates may require type-squashing

– $\downarrow P \;\equiv\; \{\mathtt{x{:}Top} \mid P\}$: reduce P to it's 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**
  - Parameterizatized, simultaneous recursion

    $\mathsf{rectype}\ X_1(x_1)\ =\ T[X_1]\ \mathsf{and}\ \ldots\ X_n(x_n)\ =\ T[X_n]\ \mathsf{select}\ X_i(a_i)$
  - Co-inductive type $\mathsf{inftype}\ X = T_X$: greatest fixed point semantics
  - Partial recursive functions $S \not\to T$: unrestricted recursive induction

Inductive Types, formally

Syntax:

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

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

Evaluation:

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

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

Semantics:

- $\mathsf{rectype}\ X_1 = T_{X1} = \mathsf{rectype}\ X_2 = T_{X2}$
  if $T_{X1}[X/X_1] = T_{X2}[X/X_2]$ for all types X
- $s = t$ in $\mathsf{rectype}\ X = T_X$ if $\mathsf{rectype}\ X = T_X$ type and
  $s = t$ in $T_X[\mathsf{rectype}\ X = T_X/X]$

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

Important Defined Types

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

See the standard theories of Nuprl 5 for further details