1 CS101 λ-calculus theory

1.1 Parents

extends Base_theory

1.2 Basic λ-calculus computational rules

declare `Cs101_lc!lambda{x. 't}` **(displayed as** $\lambda x.t$**)**
declare `Cs101_lc!apply{'a; 'b}` **(displayed as** $a\ b$**)**
![] **rewrite** `beta_reduction` {| reduce |} : $((\lambda x.t[x])\ a) \longleftrightarrow t[a]$

1.3 Structural rules

declare `Cs101_lc!member{'t; 'T}` **(displayed as** $t\ \in\ T$**)**
![·] **rule** `type_axiom` Γ:
$\langle\Gamma\rangle;\ x:\ A;\ \langle\Delta\rangle\ \vdash\ x\ \in\ A$
![·] **rule** `swap` $\Gamma\ \Delta\ \Sigma$:
$\langle\Gamma\rangle;\ \langle\Sigma\rangle;\ \langle\Delta\rangle;\ \langle L\rangle\ \vdash\ C\ \longrightarrow$
$\langle\Gamma\rangle;\ \langle\Delta\rangle;\ \langle\Sigma\rangle;\ \langle L\rangle\ \vdash\ C$
![·] **rule** `cut` C `bind`$(x.t_1[x])\ t_2$:
$\langle\Gamma\rangle\ \vdash\ t_2\ \in\ C\ \longrightarrow$
$\langle\Gamma\rangle;\ x:\ C\ \vdash\ t_1[x]\ \in\ T\ \longrightarrow$
$\langle\Gamma\rangle\ \vdash\ t_1[t_2]\ \in\ T$

1.4 Constant Types

declare `Cs101_lc!void` **(displayed as** $Void$**)**
declare `Cs101_lc!unit` **(displayed as** $Unit$**)**
![·] **rule** `void_elim` {| elim [] |} Γ:
$\langle\Gamma\rangle;\ x:\ Void;\ \langle\Delta\rangle\ \vdash\ C[x]$
![·] **rule** `unit_type` {| intro [] |} :
$\langle\Gamma\rangle\ \vdash\ \cdot\ \in\ Unit$

1.5 Function type

declare `Cs101_lc!fun{'A; x. B['x]}` **(displayed as** $x:A\ \rightarrow\ B[x]$**)**
define `unfold_fun` : `Cs101_lc!fun{'A; 'B}` $\longleftrightarrow (x:A\ \rightarrow\ B)$
![·] **rule** `lambda_dep_type` {| intro [] |} :
$\langle\Gamma\rangle;\ x:\ A\ \vdash\ t[x]\ \in\ B[x]\ \longrightarrow$
$\langle\Gamma\rangle\ \vdash\ (\lambda x.t[x])\ \in\ (x:A\ \rightarrow\ B[x])$
$*[1, 50]$ **rule** `lambda_type` {| intro [] |} :
$\langle\Gamma\rangle;\ x:\ A\ \vdash\ t[x]\ \in\ B\ \longrightarrow$

$\langle\Gamma\rangle \vdash (\lambda x.t[x]) \in (A \to B)$

$![\cdot]$ **rule** `apply_dep_type` {| intro [] |} $(x : B \to A[x])$:
$\langle\Gamma\rangle \vdash a \in (x : B \to A[x]) \longrightarrow$
$\langle\Gamma\rangle \vdash b \in B \longrightarrow$
$\langle\Gamma\rangle \vdash a\ b \in A[b]$

$*[1, 62]$ **rule** `apply_type` {| intro [] |} B:
$\langle\Gamma\rangle \vdash a \in (B \to A) \longrightarrow$
$\langle\Gamma\rangle \vdash b \in B \longrightarrow$
$\langle\Gamma\rangle \vdash a\ b \in A$

$*[1, 58]$ **rule** `example1` :
$\vdash (\lambda x.x)\ (\lambda y.y) \in (A \to A)$

$*[1, 17]$ **rule** `void_lambda` :
$\vdash (\lambda x.t[x]) \in (Void \to A)$

$*[3, 33]$ **rule** `fun_elim` Γ `bind`$(x.t_1[x])$ $Perv!bind\{x,\ y.\ t_2[x;\ y]\}$:
$\langle\Gamma\rangle;\ x :\ A \to B;\ \langle\Delta\rangle \vdash t_1[x] \in A \longrightarrow$
$\langle\Gamma\rangle;\ x :\ A \to B;\ y :\ B;\ \langle\Delta\rangle \vdash t_2[x;\ y] \in C \longrightarrow$
$\langle\Gamma\rangle;\ x :\ A \to B;\ \langle\Delta\rangle \vdash t_2[x;\ (x\ t_1[x])] \in C$

1.6 Product type

declare `Cs101_lc!pair{'a; 'b}` **(displayed as** (a, b)**)**
declare `Cs101_lc!fst{'a}` **(displayed as** $a.1$**)**
declare `Cs101_lc!snd{'a}` **(displayed as** $a.2$**)**
$![]$ **rewrite** `reduce_fst` {| reduce |} : $(a, b).1 \longleftrightarrow a$
$![]$ **rewrite** `reduce_snd` {| reduce |} : $(a, b).2 \longleftrightarrow b$
declare `Cs101_lc!prod{'A; x. B['x]}` **(displayed as** $x : A \times B[x]$**)**
define `unfold_prod` : `Cs101_lc!prod{'A; 'B}` $\longleftrightarrow (x : A \times B)$
$![\cdot]$ **rule** `dep_pair_type` {| intro [] |} :
$\langle\Gamma\rangle \vdash a \in A \longrightarrow$
$\langle\Gamma\rangle \vdash b \in B[a] \longrightarrow$
$\langle\Gamma\rangle \vdash (a, b) \in (x : A \times B[x])$

$\#$ **rule** `pair_type` {| intro [] |} :
$\langle\Gamma\rangle \vdash a \in A \longrightarrow$
$\langle\Gamma\rangle \vdash b \in B \longrightarrow$
$\langle\Gamma\rangle \vdash (a, b) \in (A \times B)$

$![\cdot]$ **rule** `pair_elim` {| elim [] |} Γ:
$\langle\Gamma\rangle;\ u :\ A;\ v :\ B;\ \langle\Delta\rangle \vdash t[(u, v)] \in C \longrightarrow$
$\langle\Gamma\rangle;\ x :\ A \times B;\ \langle\Delta\rangle \vdash t[x] \in C$

$*[2, 87]$ **rule** `fst_type` {| intro [] |} B:
$\langle\Gamma\rangle \vdash a \in (A \times B) \longrightarrow$
$\langle\Gamma\rangle \vdash a.1 \in A$

$*[2, 87]$ **rule** `snd_type` {| intro [] |} A:
$\langle\Gamma\rangle \vdash a \in (A \times B) \longrightarrow$
$\langle\Gamma\rangle \vdash a.2 \in B$

1.7 Disjoint union type

declare `Cs101_lc!inl{'x}` **(displayed as** $inl\ x$**)**
declare `Cs101_lc!inr{'x}` **(displayed as** $inr\ x$**)**
declare
`Cs101_lc!decide{'x; u. t1['u]; v. t2['v]}`
(displayed as $match\ x\ with\ inl\ u\ \rightarrow\ t_1[u]\ \mid\ inr\ v\ \rightarrow\ t_2[v]$**)**
![] **rewrite** `reduce_inl` {| reduce |} :
$match\ (inl\ x)\ with\ inl\ u\ \rightarrow\ t_1[u]\ \mid\ inr\ v\ \rightarrow\ t_2[v] \longleftrightarrow t_1[x]$
![] **rewrite** `reduce_inr` {| reduce |} :
$match\ (inr\ x)\ with\ inl\ u\ \rightarrow\ t_1[u]\ \mid\ inr\ v\ \rightarrow\ t_2[v] \longleftrightarrow t_2[x]$
declare `Cs101_lc!union{'A; 'B}` **(displayed as** $A\ +\ B$**)**
![·] **rule** `inl_type` {| intro [] |} :
$\langle\Gamma\rangle\ \vdash\ a\ \in\ A\ \longrightarrow$
$\langle\Gamma\rangle\ \vdash\ (inl\ a)\ \in\ (A\ +\ B)$
![·] **rule** `inr_type` {| intro [] |} :
$\langle\Gamma\rangle\ \vdash\ b\ \in\ B\ \longrightarrow$
$\langle\Gamma\rangle\ \vdash\ (inr\ b)\ \in\ (A\ +\ B)$
![·] **rule** `union_elim` {| elim [] |} Γ:
$\langle\Gamma\rangle;\ u:\ A;\ \langle\Delta[inl\ u]\rangle\ \vdash\ t[(inl\ u)]\ \in\ C[(inl\ u)]\ \longrightarrow$
$\langle\Gamma\rangle;\ v:\ B;\ \langle\Delta[inr\ v]\rangle\ \vdash\ t[(inr\ v)]\ \in\ C[(inr\ v)]\ \longrightarrow$
$\langle\Gamma\rangle;\ x:\ A\ +\ B;\ \langle\Delta[x]\rangle\ \vdash\ t[x]\ \in\ C[x]$
rule `decide_type` {| intro [] |} $(A\ +\ B)$:
$\langle\Gamma\rangle\ \vdash\ x\ \in\ (A\ +\ B)\ \longrightarrow$
$\langle\Gamma\rangle;\ u:\ A\ \vdash\ t_1[u]\ \in\ C\ \longrightarrow$
$\langle\Gamma\rangle;\ v:\ B\ \vdash\ t_2[v]\ \in\ C\ \longrightarrow$
$\langle\Gamma\rangle\ \vdash\ match\ x\ with\ inl\ u\ \rightarrow\ t_1[u]\ \mid\ inr\ v\ \rightarrow\ t_2[v]\ \in\ C$
$*[2, 83]$ **rule** `example2` :

$\vdash$
$(\lambda x.match\ x\ with\ inl\ u\ \rightarrow\ u\ \mid\ inr\ v\ \rightarrow\ v)$
$\in$
$(x : (A\ +\ B)\ \rightarrow\ match\ x\ with\ inl\ u\ \rightarrow\ A\ \mid\ inr\ v\ \rightarrow\ B)$

2 CS101 Intuitionistic Logic Theory

2.1 Parents

The CS101 formalization of intuitionistic logic is grounded in the Base theory, that defines basic concepts and basic proof search automation.

extends Base_theory
extends Cs101_lc

2.2 Terms

The following declarations define the language of the logic.

define `unfold_true` : `Cs101_int!true` $\longleftrightarrow Unit$
define `unfold_false` : `Cs101_int!false` $\longleftrightarrow Void$
define `unfold_or` : `Cs101_int!or{'A; 'B}` $\longleftrightarrow (A\ +\ B)$
define `unfold_and` : `Cs101_int!and{'A; 'B}` $\longleftrightarrow (A\ \times\ B)$
define `unfold_implies` : `Cs101_int!implies{'A; 'B}` $\longleftrightarrow (A\ \rightarrow\ B)$
define `unfold_not` : `Cs101_int!not{'A}` $\longleftrightarrow (A\ \rightarrow\ False)$

2.3 Rules

The rules below are exactly the same that were presented and discussed in the first week of the class.

$![t]$ **rule** `useWitness` t:
$\langle\Gamma\rangle\ \vdash\ t\ \in\ T\ \longrightarrow$
$\langle\Gamma\rangle\ \vdash\ T$
$![c[a]]$ **rule** `cutWitness` A:
a: $\langle\Gamma\rangle\ \vdash\ A\ \longrightarrow$
$c[x]$: $\langle\Gamma\rangle;\ x:\ A\ \vdash\ C\ \longrightarrow$
$\langle\Gamma\rangle\ \vdash\ C$
$*[1, 9]$ **rule** `swap` $\Gamma\ \Delta\ \Sigma$:
$\langle\Gamma\rangle;\ \langle\Sigma\rangle;\ \langle\Delta\rangle;\ \langle L\rangle\ \vdash\ C\ \longrightarrow$
$\langle\Gamma\rangle;\ \langle\Delta\rangle;\ \langle\Sigma\rangle;\ \langle L\rangle\ \vdash\ C$
$*[1, 18]$ **rule** `axiom` Γ:
$\langle\Gamma\rangle;\ A;\ \langle\Delta\rangle\ \vdash\ A$
$*[2, 20]$ **rule** `copy` Γ:
$\langle\Gamma\rangle;\ A;\ A;\ \langle\Delta\rangle\ \vdash\ C\ \longrightarrow$
$\langle\Gamma\rangle;\ A;\ \langle\Delta\rangle\ \vdash\ C$
$![t]$ **rule** `thin` Γ:
t: $\langle\Gamma\rangle;\ \langle\Delta\rangle\ \vdash\ C\ \longrightarrow$
$\langle\Gamma\rangle;\ A;\ \langle\Delta\rangle\ \vdash\ C$
$*[2, 15]$ **rule** `true_i` {| intro [] |} :
$\langle\Gamma\rangle\ \vdash\ True$
$*[1, 24]$ **rule** `false_e` {| elim [] |} Γ:
$\langle\Gamma\rangle;\ False;\ \langle\Delta\rangle\ \vdash\ C$
$![(\lambda x.t[x])]$ **rule** `imp_i` {| intro [] |} :
$t[x]$: $\langle\Gamma\rangle;\ x:\ A\ \vdash\ B\ \longrightarrow$
$\langle\Gamma\rangle\ \vdash\ A\ \rightarrow\ B$
$*[7, 112]$ **rule** `imp_e` {| elim [] |} Γ:
$\langle\Gamma\rangle;\ \langle\Delta\rangle\ \vdash\ A\ \longrightarrow$
$\langle\Gamma\rangle;\ B;\ \langle\Delta\rangle\ \vdash\ C\ \longrightarrow$
$\langle\Gamma\rangle;\ A\ \rightarrow\ B;\ \langle\Delta\rangle\ \vdash\ C$
$\#$ **rule** `or_i1` {| intro [(SelectOption 1)] |} :

$\langle \Gamma \rangle \vdash A \longrightarrow$
$\langle \Gamma \rangle \vdash A \vee B$

\# **rule** `or_i2` {| intro [(SelectOption 2)] |} :
$\langle \Gamma \rangle \vdash B \longrightarrow$
$\langle \Gamma \rangle \vdash A \vee B$

\# **rule** `or_e` {| elim [] |} Γ:
$\langle \Gamma \rangle ; A; \langle \Delta \rangle \vdash C \longrightarrow$
$\langle \Gamma \rangle ; B; \langle \Delta \rangle \vdash C \longrightarrow$
$\langle \Gamma \rangle ; A \vee B; \langle \Delta \rangle \vdash C$

\# **rule** `and_i` {| intro [] |} :
$\langle \Gamma \rangle \vdash A \longrightarrow$
$\langle \Gamma \rangle \vdash B \longrightarrow$
$\langle \Gamma \rangle \vdash A \wedge B$

\# **rule** `and_e` {| elim [] |} Γ:
$\langle \Gamma \rangle ; A; B; \langle \Delta \rangle \vdash C \longrightarrow$
$\langle \Gamma \rangle ; A \wedge B; \langle \Delta \rangle \vdash C$

\# **rule** `not_i` {| intro [] |} :
$\langle \Gamma \rangle ; A \vdash False \longrightarrow$
$\langle \Gamma \rangle \vdash \neg A$

\# **rule** `not_e` {| elim [] |} Γ:
$\langle \Gamma \rangle ; \langle \Delta \rangle \vdash A \longrightarrow$
$\langle \Gamma \rangle ; \neg A; \langle \Delta \rangle \vdash C$

3 CS101 Classical Logic Theory

The CS101 formalization of classical logic is implemented by extending the intuitionistic logic with the "proof by contradiction" rule.

extends Cs101_int
![·] **rule** `contradict` :
$\langle \Gamma \rangle ; \neg A \vdash False \longrightarrow$
$\langle \Gamma \rangle \vdash A$