Announcements

Homework 5 will be posted today (due in one week).

Midterm 1 results

Did you lose a watch?

Pick up exams from Cheryl Hickey.

Statistics
Max score: 78
Min score: 24
Average: 57
Std dev: 12

Re grade policy: Must submit request to me in writing within 2 weeks (by
Nov 1).

Extra Credit

Course grades can be improved after the semester ends in two ways:

1. A 1/3 letter grade improvement can be obtained by doing a substantial
   extra credit project (~30 hours work) during the Spring semester.

2. Larger grade improvements can be obtained by sitting in on the
course next year and turning in all homeworks and exams.

"Normal office hours/recitation schedule."

No good date for midterm. Looks like 12/20 1:30-3:30 is the best.

Exams
Rolling score: 90
Passing score: 60
"Place an exam from Cheryl Hickey."
The Simply Typed Lambda-Calculus

Operational Semantics

(⇒)
\[
\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \leftarrow \begin{array}{ll}
& t_1 \leftarrow t_2 \\
\end{array}
\]

(⇒)
\[
\text{if } \text{false} \text{ then } t_2 \text{ else } t_3 \leftarrow \begin{array}{ll}
& \text{false} \leftarrow \text{false} \\
\end{array}
\]

(⇒)
\[
\text{if } \text{true} \text{ then } t_2 \text{ else } t_3 \leftarrow \begin{array}{ll}
& \text{true} \leftarrow \text{true} \\
\end{array}
\]

(⇒)
\[
\text{if } x : T. t_2 \text{ then } t_3 \leftarrow \begin{array}{ll}
& \text{if } x : T. t_2 \leftarrow \text{if } x : T. t_2 \\
\end{array}
\]

Connections to logic

Connection to typed calculi

Extension of those simple types

Simply-typed lambda-calculus
Properties of $\lambda$-

As before, the fundamental property of the type system we have just defined is soundness with respect to the operational semantics.

1. Progress: A closed, well-typed term is not stuck.

2. Preservation: Types are preserved by one-step evaluation.

Proving Preservation

Theorem: If $\Gamma \vdash t : T$ and $t \rightarrow t'$, then $\Gamma \vdash t' : T$.

Proof: By induction on typing derivations.

[Which case is the hard one?]
Proving Preservation

Theorem: If \( \text{let } S \to \text{v} ; \text{v} \text{ where } S \neq 1 \text{ and } S \neq 1 \)

Proof: By induction on typing derivations.

\[ \text{Case } \text{AP} : \text{ Given } t = t_1 t_2 \text{ and } t_1 : t_1 \text{ and } t_2 : t_2 \]

By the inversion lemma for evaluation, there are three subcases...

\[ \text{Subcase: } t_1 = x : t_1 . t_2 \]

Show \( \text{let } S \to \text{v} ; \text{v} \text{ where } S \neq 1 \text{ and } S \neq 1 \)

\[ \text{Subcase: } t_1 = t_1 . t_1 \]

By the inversion lemma for evaluation, there are three subcases...

\[ \text{Subcase: } t_1 = x : t_1 . t_2 \]

By the inversion lemma for evaluation, there are three subcases...

\[ \text{Subcase: } t_1 = t_1 . t_1 \]

By the inversion lemma for evaluation, there are three subcases...

Uh oh.

CIS 500, 18 October 11-b
The “Substitution Lemma”

Lemma: Types are preserved under substitution.

If \( \Gamma; x : S \vdash t : T \) and \( \Gamma; s : S \vdash t' : T \), then \( \Gamma; x \rightarrow s : T \).

Proof: ...

The Unit type

\[
\begin{align*}
\text{terms} & \quad \text{values} & \quad \text{types} \\
\text{constant unit} & \quad \text{unit} & \quad \text{unit type} \\
\Gamma \vdash \text{unit : Unit} \\
\end{align*}
\]

New typing rules

On to real programming languages...
Sequencing as a derived form

\[
\text{Declarations:}
\]

- \( t_1 : \text{Unit} \)
- \( T2 : T2 \)
- \( t_1 ; t_2 : T2 \)

\[
\text{where } x \in \mathcal{F}(T2)
\]

\[
\text{E-Seq (ax:Unit.T2) \rightarrow t_1 t_2}
\]

\[
\text{E-SeqNext (T2) \rightarrow t_1 t_2}
\]

Derived forms

Syntactic sugar

- Internal language vs. external (surface) language

CIS 500, 18 October 15

CIS 500, 18 October 16

CIS 500, 18 October 17
Ascription as a derived form

\[ t \leftarrow e \]

New reduction rules

\[ \text{let } x = t \text{ in } \text{let binding} \]

New evaluation rules

\[ t \text{ as } I \text{ in } t \]

\[ [x : \tau] \tau \]

Let-bindings

New syntactic forms

\[ t \text{::= ... terms} \]

Ascription

E-Ascribe

\[ E-\text{Ascribe} \]

New typing rules

\[ t : T \]

\[ \tau \]

\[ x : T \]

\[ t_1 : T \]

\[ t \]

\[ t_2 \]

\[ t_0 \]

\[ \frac{L \vdash t \cdot \varphi \cdot t_1 \cdot t_2 \cdot t_1 : T \cdot \varphi \cdot t_2 : T}{L \vdash t} \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]

\[ L \vdash t \]
Evaluation rules for pairs

\[
\begin{align*}
\text{(E-Pair1)} & : L \vdash \{v_1, v_2\} : T_1 \times T_2 \\
\text{(E-Pair2)} & : L \vdash \{v_1, v_2\} : T_1 \\
\text{(E-Projection1)} & : L \vdash \{v_1, v_2\} : T_1 \\
\text{(E-Projection2)} & : L \vdash \{v_1, v_2\} : T_1 \\
\end{align*}
\]

Typing rules for pairs

\[
\begin{align*}
\text{(T-Pair)} & : T : L_1, L_2 \vdash \{v_1, v_2\} : T_1 \times T_2 \\
\text{(T-Proj1)} & : T : L_1, L_2 \vdash v_2 : T_2 \\
\text{(T-Proj2)} & : T : L_1, L_2 \vdash v_1 : T_1 \\
\end{align*}
\]

Tuples

\[
\begin{align*}
\text{tuple type} & : \{u_1, \ldots, u_n\} :: \tau \\
\text{tuple value} & : \{u_1, \ldots, u_n\} :: \tau \\
\text{tuple} & : \{u_1, \ldots, u_n\} :: \tau \\
\text{projection} & : \{u_1, \ldots, u_n\} :: \tau \\
\text{terms} & : \{u_1, \ldots, u_n\} :: \tau \\
\text{types} & : \{u_1, \ldots, u_n\} :: \tau \\
\end{align*}
\]
Evaluation rules for tuples

\[ \{ u \cdot t \in T \mid t \in \text{Var}, t \neq \text{Var} \} \leftarrow \{ u \cdot t \in T \mid t \in \text{Var}, t \neq \text{Var} \} \]

\[ j \leftarrow t \]

Typing rules for tuples

\[ \{ u \cdot t \in T \mid t \in \text{Var}, t \neq \text{Var} \} \leftarrow \{ u \cdot t \in T \mid t \in \text{Var}, t \neq \text{Var} \} \]

\[ \{ u \cdot t \in T \mid t \in \text{Var}, t \neq \text{Var} \} \leftarrow \{ u \cdot t \in T \mid t \in \text{Var}, t \neq \text{Var} \} \]

\[ \text{for each } t \in \text{Var}, t \neq \text{Var} \]

\[ j \leftarrow t \]

Evaluation rules for records

\[ \{ u \cdot t \in T \mid t \in \text{Var}, t \neq \text{Var} \} \leftarrow \{ u \cdot t \in T \mid t \in \text{Var}, t \neq \text{Var} \} \]

\[ j \leftarrow t \]

Records

\[ \{ u \cdot t \in T \mid t \in \text{Var}, t \neq \text{Var} \} \leftarrow \{ u \cdot t \in T \mid t \in \text{Var}, t \neq \text{Var} \} \]

\[ j \leftarrow t \]
Typing rules for records:

For each $i$:

\[ T_i : \{ l_i = t_i : T_i \} \]

Connection with untyped lambda calculus:

Erasure:

\[
\begin{align*}
\text{erase} (x) &= x \\
\text{erase} (x:T_1 . t_2) &= \text{erase} (x) . \text{erase} (t_2) \\
\text{erase} (t_1 t_2) &= \text{erase} (t_1) \cdot \text{erase} (t_2)
\end{align*}
\]

Theorem:

1. If $t \vdash t_0$ then $\text{erase} (t) \vdash \text{erase} (t_0)$.
2. If $\text{erase} (t) \not\vdash m_0$, then there is a simply typed term $c$, such that $c \vdash t$.

Intro vs. Elim forms:

When laying out introduction forms, what are elimination forms?

An elimination form for a type gives us a way of using elements of this type.

An introduction form for a given type gives us a way of constructing elements.

Typing rules for records:

\[
\frac{T \vdash t : T}{L \vdash \{ l : T \}}
\]

\[
\frac{T \vdash t \text{ for each } i}{L \vdash \{ \forall i. l_i : T_i \}}
\]
An untyped term $m$ is said to be typable if there is some context $\Gamma$ in the simply typed lambda-calculus, some type $T$, and some context $L$ such that 

\[ \Gamma \vdash m : T \]

Typability

Typability
Propositions as Types

Logic Programming Languages

Propositions

Types

Proofs

Simplification

Evaluation

Propositions types

Proposition

P

Q

Type

P

→

Q

Proposition

true

Type

unit

Proof of proposition

P

Term

t

of type

P

Proposition

P

is provable

Type

P

is inhabited (by some term)

Proofs simplification (a.k.a. cut elimination)