Substructural Type Systems

CS 152 (Spring 2024)

Harvard University

Tuesday, March 26, 2024

Today

- Substructural type systems

But first, some code

```
int foo() {
  int *scratch = malloc(sizeof(int) * 10);
  if (error) {

    return 1;
  }
  // do some computation
  free(scratch);
  return result;
}
```

But first, some code

```
int foo() {
  int *scratch = malloc(sizeof(int) * 10);
  if (error) {
    // Oops! forgot to free scratch!
    return 1;
  }
  // do some computation
  free(scratch);
  return result;
}
```

But first, some code

```c
int *bar() {
  int *x = malloc(sizeof(int));
  int *y = x;

  free(y);

  return *x;
}
```

But first, some code

```
int *bar() {
  int *x = malloc(sizeof(int));
  int *y = x;

  free(y);
  // Pointer to freed memory still alive!
  return *x;
}
```

- Not limited to memory management!
  - File handles
  - Network connections
  - Many more...

Driving Question

- How can we use logic to represent *resources*?

Natural deduction

- *Natural deduction* is a kind of proof calculus that can be used to formalize mathematical logic
  - Meant to be the natural way to reason about truth!
- $A_1, \ldots, A_n \vdash B$ means whenever formulas A_1 to A_n are true, then formula B is true
- E.g., $p, \neg q \vdash q \Rightarrow (p \Rightarrow r)$
- Can define inference rules for the logic, e.g.,

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}$$

Natural deduction inference rules (propositional logic)

$$\frac{}{A \vdash A} \qquad \frac{\Gamma, B \vdash A}{\Gamma \vdash B \Rightarrow A}$$

$$\frac{\Gamma \vdash B \Rightarrow A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A} \qquad \frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \wedge B}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$$

Structural inference rules

- *Structural inference rules* manipulate the assumptions (i.e., formulas to the left of $\vdash$)
- Allow us to treat list of formulas like a set.

Structural inference rule (1/3)

$$\textsc{Exchange}\ \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}$$

Structural inference rule (2/3)

$$\textsc{Contraction}\ \frac{\Gamma, A, A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}$$

Structural inference rule (3/3)

$$\textsc{Weakening}\ \frac{\Gamma, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}$$

Substructural logics

- If we drop any structural inference rule, we have a *substructural logic*

Substructural logics: affine logic

- Keep Exchange and Weakening, drop Contraction
  - Every assumption must be used at most once

$$\textsc{Exchange}\ \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}$$

$$\textsc{Contraction}\ \frac{\Gamma, A, A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}$$

$$\textsc{Weakening}\ \frac{\Gamma, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}$$

Substructural logics: linear logic

- Keep Exchange but drop Weakening and Contraction: **linear logic**
  - Every assumption must be used exactly once

$$\textsc{Exchange}\ \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}$$

$$\textsc{Contraction}\ \frac{\Gamma, A, A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}$$

$$\textsc{Weakening}\ \frac{\Gamma, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}$$

Curry-Howard Correspondence

Curry-Howard Correspondence

- Exchange

$$\frac{\Gamma, x:\tau_1, y:\tau_2, \Delta \vdash e:\tau}{\Gamma, y:\tau_2, x:\tau_1, \Delta \vdash e:\tau} \qquad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}$$

- Contraction

$$\frac{\Gamma, x:\tau, x:\tau, \Delta \vdash e:\tau'}{\Gamma, x:\tau, \Delta \vdash e:\tau'} \qquad \frac{\Gamma, A, A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}$$

- Weakening

$$\frac{\Gamma, \Delta \vdash e:\tau}{\Gamma, x:\tau', \Delta \vdash e:\tau}\, x \notin \Gamma, \Delta \qquad \frac{\Gamma, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}$$

Affine type system

- Every variable is used at most once
- Affine type systems drop Contraction (but keep Exchange and Weakening)

$$\textsc{Exchange}\ \frac{\Gamma, x : \tau_1, y : \tau_2, \Delta \vdash e : \tau}{\Gamma, y : \tau_2, x : \tau_1, \Delta \vdash e : \tau}$$

$$\textsc{Contraction}\ \frac{\Gamma, x : \tau, x : \tau, \Delta \vdash e : \tau'}{\Gamma, x : \tau, \Delta \vdash e : \tau'}$$

$$\textsc{Weakening}\ \frac{\Gamma, \Delta \vdash e : \tau}{\Gamma, x : \tau', \Delta \vdash e : \tau}\ x \text{ not in } \Gamma, \Delta$$

Relevant type system

- Every variable is used at least once
- Relevant type systems drop Weakening (but keep Contraction and Exchange)

$$\textsc{Exchange}\ \frac{\Gamma, x{:}\tau_1, y{:}\tau_2, \Delta \vdash e{:}\tau}{\Gamma, y{:}\tau_2, x{:}\tau_1, \Delta \vdash e{:}\tau}$$

$$\textsc{Contraction}\ \frac{\Gamma, x{:}\tau, x{:}\tau, \Delta \vdash e{:}\tau'}{\Gamma, x{:}\tau, \Delta \vdash e{:}\tau'}$$

$$\textsc{Weakening}\ \frac{\Gamma, \Delta \vdash e{:}\tau}{\Gamma, x{:}\tau', \Delta \vdash e{:}\tau}\ x \text{ not in } \Gamma, \Delta$$

Linear type system

- Every variable is used exactly once
- Linear type systems drop Contraction and Weakening (but keep Exchange)

$$\textsc{Exchange}\ \frac{\Gamma, x : \tau_1, y : \tau_2, \Delta \vdash e : \tau}{\Gamma, y : \tau_2, x : \tau_1, \Delta \vdash e : \tau}$$

$$\textsc{Contraction}\ \frac{\Gamma, x : \tau, x : \tau, \Delta \vdash e : \tau'}{\Gamma, x : \tau, \Delta \vdash e : \tau'}$$

$$\textsc{Weakening}\ \frac{\Gamma, \Delta \vdash e : \tau}{\Gamma, x : \tau', \Delta \vdash e : \tau}\ x \text{ not in } \Gamma, \Delta$$

Ordered type system

- Every variable is used exactly once, in order
- Ordered type systems drop Weakening, Contraction, and Exchange

$$\textsc{Exchange}\ \frac{\Gamma, x{:}\tau_1, y{:}\tau_2, \Delta \vdash e{:}\tau}{\Gamma, y{:}\tau_2, x{:}\tau_1, \Delta \vdash e{:}\tau}$$

$$\textsc{Contraction}\ \frac{\Gamma, x{:}\tau, x{:}\tau, \Delta \vdash e{:}\tau'}{\Gamma, x{:}\tau, \Delta \vdash e{:}\tau'}$$

$$\textsc{Weakening}\ \frac{\Gamma, \Delta \vdash e{:}\tau}{\Gamma, x{:}\tau', \Delta \vdash e{:}\tau}\ x \text{ not in } \Gamma, \Delta$$

Linear lambda calculus

- Explore linear type system in lambda calculus
- Type system will track use of objects
- A *linear object* must be used exactly once (and implementation could, e.g., deallocate object after use)
- Will also have unrestricted objects that can be used many times

Syntax

$$q ::= \mathsf{lin} \mid \mathsf{un}$$

$$\begin{aligned} e ::= {} & x \mid q\ b \mid q\ (e_1, e_2) \mid q\ \lambda x{:}\tau.\, e \mid e_1\ e_2 \\ & \mid \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 \mid \mathsf{split}\ e_1\ \mathsf{as}\ x, y\ \mathsf{in}\ e_2 \end{aligned}$$

$$b \in \{\mathsf{true}, \mathsf{false}\}$$

Type system

$$\pi ::= \textbf{bool} \mid \tau_1 \times \tau_2 \mid \tau_1 \to \tau_2$$

$$\tau ::= q\ \pi$$

$$\Gamma ::= \emptyset \mid \Gamma, x : \tau$$

Inference rules

- Maintain two invariants:
  1. linear variables are used exactly once on each control flow path
  2. unrestricted data structures may not contain linear data structures

Utility functions (1/2)

- Split context Γ into two pieces

$$\frac{}{\emptyset = \emptyset \circ \emptyset}$$

$$\frac{\Gamma = \Gamma_1 \circ \Gamma_2}{\Gamma, x : \mathsf{un}\ \pi = (\Gamma_1, x : \mathsf{un}\ \pi) \circ (\Gamma_2, x : \mathsf{un}\ \pi)}$$

$$\frac{\Gamma = \Gamma_1 \circ \Gamma_2}{\Gamma, x : \mathsf{lin}\ \pi = (\Gamma_1, x : \mathsf{lin}\ \pi) \circ \Gamma_2}$$

$$\frac{\Gamma = \Gamma_1 \circ \Gamma_2}{\Gamma, x : \mathsf{lin}\ \pi = \Gamma_1 \circ (\Gamma_2, x : \mathsf{lin}\ \pi)}$$

Utility functions (2/2)

- Determine whether type or context can be used in linear setting
  - $\text{un}(\tau)$ if and only if $\tau = \text{un } \pi$.
  - $\text{lin}(\tau)$ if and only if $\tau = \text{un } \pi$ or $\tau = \text{lin } \pi$.
  - $q(\Gamma)$ if and only if for all $(x{:}\tau) \in \Gamma$, we have $q(\tau)$.

Inference rules (1/2)

$$\frac{\mathrm{un}(\Gamma_1, \Gamma_2)}{\Gamma_1, x : \tau, \Gamma_2 \vdash x : \tau} \qquad \frac{\mathrm{un}(\Gamma)}{\Gamma \vdash q\ b : q\ \mathbf{bool}}$$

$$\frac{\begin{array}{c}\Gamma_1 \vdash e_1 : q\ \mathbf{bool} \\ \Gamma_2 \vdash e_2 : \tau \qquad \Gamma_2 \vdash e_3 : \tau\end{array}}{\Gamma \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : \tau}\ \Gamma = \Gamma_1 \circ \Gamma_2$$

$$\frac{\Gamma_1 \vdash e_1 : \tau_1 \quad \Gamma_2 \vdash e_2 : \tau_2 \quad q(\tau_1) \quad q(\tau_2)}{\Gamma \vdash q\ (e_1, e_2) : q\ (\tau_1, \tau_2)}\ \Gamma = \Gamma_1 \circ \Gamma_2$$

Inference rules (2/2)

$$\frac{\Gamma_1 \vdash e_1 : q\ (\tau_1 \times \tau_2) \quad \Gamma_2, x : \tau_1, y : \tau_2 \vdash e_2 : \tau}{\Gamma \vdash \mathsf{split}\ e_1\ \mathsf{as}\ x, y\ \mathsf{in}\ e_2 : \tau}\ \Gamma = \Gamma_1 \circ \Gamma_2$$

$$\frac{q(\Gamma) \qquad \Gamma, x : \tau \vdash e : \tau'}{\Gamma \vdash q\ \lambda x : \tau .\ e : q\ \tau \to \tau'}$$

$$\frac{\Gamma_1 \vdash e_1 : q\ \tau \to \tau' \qquad \Gamma_2 \vdash e_2 : \tau}{\Gamma \vdash e_1\ e_2 : \tau'}\ \Gamma = \Gamma_1 \circ \Gamma_2$$

Example 1

$$
\begin{array}{l}
\mathsf{lin}\ \lambda x\!:\!\mathsf{lin}\ \textbf{bool}.\\
\qquad (\mathsf{lin}\ \lambda f\!:\!\mathsf{un}\ (\mathsf{un}\ \textbf{bool} \rightarrow \mathsf{lin}\ \textbf{bool}).\, \mathsf{lin}\ \mathsf{true})\\
\qquad (\mathsf{un}\ \lambda y\!:\!\mathsf{un}\ \textbf{bool}.\, x)
\end{array}
$$

Example 2

$$\begin{aligned}
&\mathsf{lin}\ \lambda x\!:\!\mathsf{lin}\ \mathbf{bool}.\\
&\qquad(\mathsf{lin}\ \lambda f\!:\!\mathsf{un}\ (\mathsf{un}\ \mathbf{bool} \to \mathsf{lin}\ \mathbf{bool}).\ \mathsf{lin}\ (f\ (\mathsf{un}\ \mathsf{true}), f\ (\mathsf{un}\ \mathsf{true})))\\
&\qquad(\mathsf{un}\ \lambda y\!:\!\mathsf{un}\ \mathbf{bool}.\ x)
\end{aligned}$$

Operational semantics

- Use store-based semantics (to emphasize reclaiming memory)
- Type system ensures that a location is never accessed after it is freed.

$$
\begin{aligned}
p &::= b \mid \lambda x : \tau .\, e \mid (\ell_1, \ell_2) \\
v &::= q\ p \\
E &::= [\cdot] \mid \text{if } E \text{ then } e_2 \text{ else } e_3 \mid q\ (E, e) \mid q\ (\ell, E) \\
&\quad \mid \text{split } E \text{ as } x, y \text{ in } e \mid E\ e \mid \ell\ E
\end{aligned}
$$

$$
\frac{< e, \sigma > \longrightarrow < e', \sigma' >}{< E[e], \sigma > \longrightarrow < E[e'], \sigma' >}
$$

$$\textsc{Val}\ \frac{}{\langle v, \sigma \rangle \longrightarrow \langle \ell, \sigma[\ell \mapsto v] \rangle}\ \ell \notin \mathsf{dom}(\sigma)$$

$$\textsc{If-True}\ \frac{\sigma(\ell) = q\ \mathsf{true} \qquad \sigma' = \begin{cases} \sigma & \text{if } q = \mathsf{un} \\ \sigma \setminus \ell & \text{if } q = \mathsf{lin} \end{cases}}{\langle \mathsf{if}\ \ell\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2, \sigma \rangle \longrightarrow \langle e_1, \sigma' \rangle}$$

$$\textsc{If-False}\ \frac{\sigma(\ell) = q\ \mathsf{false} \qquad \sigma' = \begin{cases} \sigma & \text{if } q = \mathsf{un} \\ \sigma \setminus \ell & \text{if } q = \mathsf{lin} \end{cases}}{\langle \mathsf{if}\ \ell\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2, \sigma \rangle \longrightarrow \langle e_2, \sigma' \rangle}$$

$$\textsc{Split}\ \frac{\sigma(\ell) = q\ (\ell_1, \ell_2) \qquad \sigma' = \begin{cases} \sigma & \text{if } q = \mathsf{un} \\ \sigma \setminus \ell & \text{if } q = \mathsf{lin} \end{cases}}{\langle \mathsf{split}\,\ell\,\mathsf{as}\,x, y\,\mathsf{in}\,e, \sigma\rangle \longrightarrow \langle e\{\ell_1/x\}\{\ell_2/y\}, \sigma'\rangle}$$

$$\textsc{App}\ \frac{\sigma(\ell_1) = q\ \lambda x\!:\!\tau.\, e \qquad \sigma' = \begin{cases} \sigma & \text{if } q = \mathsf{un} \\ \sigma \setminus \ell_1 & \text{if } q = \mathsf{lin} \end{cases}}{\langle \ell_1\ \ell_2, \sigma\rangle \longrightarrow \langle e\{\ell_2/x\}, \sigma'\rangle}$$

Type Soundness

If $\vdash e : \tau$ and $\langle e, \emptyset \rangle \longrightarrow^{*} \langle e', \sigma \rangle$ then either

- e' is not stuck or
- $\exists l$ such that $e' = l$ and $\forall l' \in \text{dom}(\sigma)$ and $\notin \text{reachable}(l, \sigma)$, $\exists p$ such that $\sigma(l') = \text{un } p$.

Substructural type systems in the wild