Today, we will learn about

- Substructural type systems
  - Natural deduction inference rules
  - Structural inference rules
  - Linear lambda calculus

- Rust
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.,

\[
\begin{align*}
\Gamma \vdash A \land B \\
\hline
\Gamma \vdash A
\end{align*}
\]
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)

\[
\text{EXCHANGE} \quad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}
\]
Structural inference rule (2/3)

\[ \text{CONTRACTION} \quad \frac{\Gamma, A, A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B} \]
Weakening

\[
\begin{align*}
\text{WEAKENING} & \quad \Gamma, \Delta \vdash B \\
& \Rightarrow \\
\Gamma, A, \Delta \vdash B
\end{align*}
\]
Natural deduction inference rules

- Additional inference rules needed, e.g., inference rules for propositional logic:

\[
\begin{align*}
\Gamma \vdash A & \quad \Gamma, B \vdash A \\
A \vdash A & \quad \Gamma \vdash B \Rightarrow A
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash B \Rightarrow A & \quad \Delta \vdash B \\
\Gamma, \Delta \vdash A
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash A & \quad \Delta \vdash B \\
\Gamma, \Delta \vdash A \land B
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash A \land B & \\
\Gamma \vdash A
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash A \land B & \\
\Gamma \vdash B
\end{align*}
\]
Substructural logics

- If we drop any structural inference rule, we have a *substructural logic*
Substructural logics: linear logic

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

\[
\text{Exchange} \quad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}
\]

\[
\text{Contraction} \quad \frac{\Gamma, A, A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}
\]

\[
\text{Weakening} \quad \frac{\Gamma, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}
\]
Substructural logics: affine logic

- Keep Exchange and Weakening but Contraction: **affine logic**
  - Every assumption must be used at most once

\[
\text{Exchange} \quad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}
\]

\[
\text{Contraction} \quad \frac{\Gamma, A, A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}
\]

\[
\text{Weakening} \quad \frac{\Gamma, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}
\]
Curry-Howard Isomorphism
Curry-Howard Isomorphism

- **Exchange**
  \[
  \Gamma, x : \tau_1, y : \tau_2, \Delta \vdash e : \tau \quad \Rightarrow \\
  \Gamma, y : \tau_2, x : \tau_1, \Delta \vdash e : \tau
  \]
  \[
  \Gamma, A, B, \Delta \vdash C \\
  \Gamma, B, A, \Delta \vdash C
  \]

- **Contraction**
  \[
  \Gamma, x : \tau, x : \tau, \Delta \vdash e : \tau' \\
  \Gamma, x : \tau, \Delta \vdash e : \tau'
  \]
  \[
  \Gamma, A, A, \Delta \vdash B \\
  \Gamma, A, \Delta \vdash B
  \]

- **Weakening**
  \[
  \Gamma, \Delta \vdash e : \tau
  \]
  \[
  \Gamma, x : \tau', \Delta \vdash e : \tau \\
  x \notin \Gamma, \Delta
  \]
  \[
  \Gamma, \Delta \vdash B \\
  \Gamma, A, \Delta \vdash B
  \]
Linear type system

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

**Exchange**

\[
\frac{\Gamma, x : \tau_1, y : \tau_2, \Delta \vdash e : \tau}{\Gamma, y : \tau_2, x : \tau_1, \Delta \vdash e : \tau}
\]

**Contraction**

\[
\frac{\Gamma, x : \tau, x : \tau, \Delta \vdash e : \tau'}{\Gamma, x : \tau, \Delta \vdash e : \tau'}
\]

**Weakening**

\[
\frac{\Gamma, \Delta \vdash e : \tau}{\Gamma, x : \tau', \Delta \vdash e : \tau} \quad x \text{ not in } \Gamma, \Delta
\]
Affine type system

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

**Exchange**

\[
\frac{\Gamma, x : \tau_1, y : \tau_2, \Delta \vdash e : \tau}{\Gamma, y : \tau_2, x : \tau_1, \Delta \vdash e : \tau}
\]

**Contraction**

\[
\frac{\Gamma, x : \tau, x : \tau, \Delta \vdash e : \tau'}{\Gamma, x : \tau, \Delta \vdash e : \tau'}
\]

**Weakening**

\[
\frac{\Gamma, \Delta \vdash e : \tau}{\Gamma, x : \tau', \Delta \vdash e : \tau}
\quad \text{x not in } \Gamma, \Delta
\]
Relevant type system

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

**Exchange**

\[
\frac{\Gamma, x : \tau_1, y : \tau_2, \Delta \vdash e : \tau}{\Gamma, y : \tau_2, x : \tau_1, \Delta \vdash e : \tau}
\]

**Contraction**

\[
\frac{\Gamma, x : \tau, x : \tau, \Delta \vdash e : \tau'}{\Gamma, x : \tau, \Delta \vdash e : \tau'}
\]

**Weakening**

\[
\frac{\Gamma, \Delta \vdash e : \tau}{\Gamma, x : \tau', \Delta \vdash e : \tau \quad x \text{ not in } \Gamma, \Delta}
\]
Ordered type system

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

\[
\begin{align*}
\text{Exchange} & : & \Gamma, x : \tau_1, y : \tau_2, \Delta \vdash e : \tau \\
& & \Gamma, y : \tau_2, x : \tau_1, \Delta \vdash e : \tau \\
\text{Contraction} & : & \Gamma, x : \tau, x : \tau, \Delta \vdash e : \tau' \\
& & \Gamma, x : \tau, \Delta \vdash e : \tau' \\
\text{Weakening} & : & \Gamma, \Delta \vdash e : \tau \\
& & \Gamma, x : \tau', \Delta \vdash e : \tau \\
& & x \ \text{not in } \Gamma, \Delta
\end{align*}
\]
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

\[ q ::= \text{lin} \mid \text{un} \]
\[ e ::= x \mid q \ b \mid q \ (e_1, e_2) \mid q \ \lambda x : \tau. \ e \mid e_1 \ e_2 \]
\[ \mid \text{if } e_1 \ \text{then } e_2 \ \text{else } e_3 \mid \text{split } e_1 \ \text{as } x, y \ \text{in } e_2 \]
\[ b \in \{ \text{true, false} \} \]
Type system

\[
\pi ::= \text{bool} \mid \tau_1 \times \tau_2 \mid \tau_1 \rightarrow \tau_2 \\
\tau ::= q \ \pi \\
\Gamma ::= \emptyset \mid \Gamma, x : \tau
\]
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 $\Gamma$ into two pieces

\[
\emptyset = \emptyset \circ \emptyset
\]

\[
\Gamma = \Gamma_1 \circ \Gamma_2
\]

\[
\Gamma, x : \text{un } \pi = (\Gamma_1, x : \text{un } \pi) \circ (\Gamma_2, x : \text{un } \pi)
\]

\[
\Gamma = \Gamma_1 \circ \Gamma_2
\]

\[
\Gamma, x : \text{lin } \pi = (\Gamma_1, x : \text{lin } \pi) \circ \Gamma_2
\]

\[
\Gamma = \Gamma_1 \circ \Gamma_2
\]

\[
\Gamma, x : \text{lin } \pi = \Gamma_1 \circ (\Gamma_2, x : \text{lin } \pi)
\]
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)

\[
\frac{\text{un}(\Gamma_1, \Gamma_2)}{\Gamma_1, x:\tau, \Gamma_2 \vdash x:\tau}
\]

\[
\frac{\text{un}(\Gamma)}{\Gamma \vdash q \ b: q \ \text{bool}}
\]

\[
\frac{\Gamma_1 \vdash e_1: q \ \text{bool}}{\Gamma_2 \vdash e_2: \tau \quad \Gamma_2 \vdash e_3: \tau}
\]

\[
\frac{\Gamma \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3: \tau}{\Gamma = \Gamma_1 \circ \Gamma_2}
\]

\[
\frac{\Gamma_1 \vdash e_1: \tau_1 \quad \Gamma_2 \vdash e_2: \tau_2}{\Gamma \vdash q \ (e_1, e_2): q \ (\tau_1, \tau_2)}
\]

\[
\Gamma = \Gamma_1 \circ \Gamma_2
\]
Inference rules (2/2)

\[
\frac{
\Gamma_1 \vdash e_1 : q \left( \tau_1 \times \tau_2 \right) \quad \Gamma_2, x : \tau_1, y : \tau_2 \vdash e_2 : \tau 
}{
\Gamma \vdash \text{split } e_1 \text{ as } x, y \text{ in } e_2 : \tau 
}\]

\[
\frac{
q(\Gamma) \quad \Gamma, x : \tau \vdash e : \tau'
}{
\Gamma \vdash q \lambda x : \tau. e : q \left( \tau \rightarrow \tau' \right)
}\]

\[
\frac{
\Gamma_1 \vdash e_1 : q \left( \tau \rightarrow \tau' \right) \quad \Gamma_2 \vdash e_2 : \tau 
}{
\Gamma \vdash e_1 \ e_2 : \tau'
}\]

\[
\Gamma = \Gamma_1 \circ \Gamma_2
\]
Examples

\[ \text{lin } \lambda x : \text{lin } \text{bool}. \]
\[ (\text{lin } \lambda f : \text{un} \ (\text{un } \text{bool} \rightarrow \text{lin } \text{bool}). \text{lin } \text{true}) \]
\[ (\text{un } \lambda y : \text{un } \text{bool}. \ x) \]

\[ \text{lin } \lambda x : \text{lin } \text{bool}. \]
\[ (\text{lin } \lambda f : \text{un} \ (\text{un } \text{bool} \rightarrow \text{lin } \text{bool}). \text{lin } (f \ (\text{un } \text{true}), f \ (\text{un } \text{true}))) \]
\[ (\text{un } \lambda y : \text{un } \text{bool}. \ x) \]
Operational semantics

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

\[
p ::= b \mid \lambda x:\tau. e \mid (\ell_1, \ell_2)
\]

\[
v ::= q \ p
\]

\[
E ::= \lfloor \cdot \rfloor \mid \text{if } E \text{ then } e_2 \text{ else } e_3 \mid q (E, e) \mid q (\ell, E)
\]

\[
\mid \text{split } E \text{ as } x, y \text{ in } e \mid E \ e \mid \ell \ E
\]

\[
< e, \sigma > \xrightarrow{} < e', \sigma' >
\]

\[
< E[e], \sigma > \xrightarrow{} < E[e'], \sigma' >
\]
\[ \text{VAL} \quad \frac{\langle v, \sigma \rangle \longrightarrow \langle \ell, \sigma[\ell \mapsto v] \rangle}{\ell \notin \text{dom}(\sigma)} \]

\[ \text{IF-TRUE} \quad \frac{\sigma(\ell) = q \text{ true}}{\langle \text{if } \ell \text{ then } e_1 \text{ else } e_2, \sigma \rangle \longrightarrow \langle e_1, \sigma' \rangle} \]

\[ \sigma(\ell) = q \text{ true} \quad \sigma' = \begin{cases} \sigma & \text{if } q = \text{un} \\ \sigma \setminus \ell & \text{if } q = \text{lin} \end{cases} \]

\[ \text{IF-FALSE} \quad \frac{\sigma(\ell) = q \text{ false}}{\langle \text{if } \ell \text{ then } e_1 \text{ else } e_2, \sigma \rangle \longrightarrow \langle e_2, \sigma' \rangle} \]

\[ \sigma(\ell) = q \text{ false} \quad \sigma' = \begin{cases} \sigma & \text{if } q = \text{un} \\ \sigma \setminus \ell & \text{if } q = \text{lin} \end{cases} \]
\[
\begin{align*}
\text{SPLIT} & \quad \sigma(\ell) = q (\ell_1, \ell_2) \quad \sigma' = \begin{cases} 
\sigma & \text{if } q = \text{un} \\
\sigma \setminus \ell & \text{if } q = \text{lin}
\end{cases} \\
\langle \text{split } \ell \text{ as } x, y \text{ in } e, \sigma \rangle & \quad \rightarrow \quad \langle e\{\ell_1/x\}\{\ell_2/y\}, \sigma' \rangle
\end{align*}
\]

\[
\begin{align*}
\text{APP} & \quad \sigma(\ell_1) = q \lambda x : \tau. e \quad \sigma' = \begin{cases} 
\sigma & \text{if } q = \text{un} \\
\sigma \setminus \ell_1 & \text{if } q = \text{lin}
\end{cases} \\
\langle \ell_1 \ell_2, \sigma \rangle & \quad \rightarrow \quad \langle e\{\ell_2/x\}, \sigma' \rangle
\end{align*}
\]
Type Soundness

If \( \vdash e : \tau \) and \( \langle e, \emptyset \rangle \rightarrow \langle e', \sigma \rangle \) then either

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