Stratifications and complexity in linear logic

Daniel Murfet
### Curry-Howard correspondence

<table>
<thead>
<tr>
<th>logic</th>
<th>programming</th>
<th>categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>formula</td>
<td>type</td>
<td>objects</td>
</tr>
<tr>
<td>sequent</td>
<td>input/output spec</td>
<td>—</td>
</tr>
<tr>
<td>proof</td>
<td>program</td>
<td>morphisms</td>
</tr>
<tr>
<td>cut-elimination</td>
<td>execution</td>
<td>—</td>
</tr>
<tr>
<td>contraction</td>
<td>copying</td>
<td>coproducts</td>
</tr>
<tr>
<td>stratification</td>
<td>complexity</td>
<td>?</td>
</tr>
</tbody>
</table>

**this talk**
Outline

1. Turing machines
2. Sequent calculus of linear logic
3. Programs (Turing machines) as proofs
4. Stratification vs. complexity
A configuration is \((w_L, w_R, a) \in \{0, 1\}^* \times \{0, 1\}^* \times \{1, \ldots, q\}\)

Shown configuration: \((1011 \cdots, 101 \cdots, 3)\)
Turing machines

A Turing machine $T$ is a function

$$\delta_T : \{0, 1\} \times \{1, \ldots, q\} \longrightarrow \{0, 1\} \times \{1, \ldots, q\} \times \{L, R\}$$

<table>
<thead>
<tr>
<th>read symbol</th>
<th>current state</th>
<th>write</th>
<th>new state</th>
<th>move to</th>
</tr>
</thead>
</table>

$$\{0, 1\}^* \times \{0, 1\}^* \times \{1, \ldots, q\} \longrightarrow \{0, 1\}^* \times \{0, 1\}^* \times \{1, \ldots, q\}$$
Linear logic

• Discovered by Girard in the 1980s, linear logic is a substructural logic with contraction and weakening available only for formulas marked with an “exponential” connective, written “!”. 

• The usual connectives of logic (e.g. conjunction, implication) are decomposed into ! together with a linearised version of that connective (called resp. tensor, linear implication).

• Under Curry-Howard, linear logic corresponds to a programming language with “resource management” and symmetric monoidal categories equipped with a special kind of comonad.

• We will use second-order intuitionistic linear logic with additives (as expressive as polymorphic lambda calculus).
linear logic

variables: $\alpha, \beta, \gamma, \ldots$

formulas: $!F, F \otimes F', F \multimap F', F \& F', \forall \alpha F$, constants

\[
\text{int} = \forall \alpha !(\alpha \multimap \alpha) \multimap (\alpha \multimap \alpha)
\]

\[
\text{bint} = \forall \alpha !(\alpha \multimap \alpha) \multimap (! (\alpha \multimap \alpha) \multimap (\alpha \multimap \alpha))
\]
Deduction rules for linear logic

(Axiom): \( \frac{}{A \vdash A} \)  
(Cut): \( \frac{\Gamma \vdash A \quad \Delta', A, \Delta \vdash B}{\Delta', \Gamma, \Delta \vdash B} \) \(_{\text{cut}}\)  
(Exchange): \( \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C} \)  

(Left \( \otimes \)): \( \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \otimes B, \Delta \vdash C} \) \(_{\otimes L}\)  
(Right \( \otimes \)): \( \frac{\Gamma \vdash A}{\Gamma, \Delta \vdash A \otimes B} \) \(_{\otimes R}\)  

(Right \( \rhd \)): \( \frac{A, \Gamma \vdash B}{\Gamma \vdash A \rhd B} \) \(_{\rhd R}\)  
(Left \( \rhd \)): \( \frac{\Gamma \vdash A \quad \Delta', B, \Delta \vdash C}{\Delta', \Gamma, A \rhd B, \Delta \vdash C} \) \(_{\rhd L}\)  

(Promotion): \( \frac{\Gamma \vdash A}{\Gamma \vdash !A} \) \(_{\text{prom}}\)  
(Dereliction): \( \frac{\Gamma, A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B} \) \(_{\text{der}}\)  
(Weakening): \( \frac{\Gamma, !A, \Delta \vdash B}{\Gamma \vdash !A, \Delta \vdash B} \) \(_{\text{weak}}\)  

(Contraction): \( \frac{\Gamma, !A, !A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B} \) \(_{\text{ctr}}\)  
(Left \( \forall \)): \( \frac{\Gamma \vdash A}{\Gamma, \forall x. A \vdash C} \) \(_{\forall L}\)  
(Right \( \forall \)): \( \frac{\Gamma \vdash A}{\Gamma \vdash \forall x. A} \) \(_{\forall R}\)  

a sequent is \( \Gamma \vdash A \) for a sequence of formulae \( \Gamma \), where \( \vdash \) is the “turnstile”
Binary integers

\[ \text{bint} = \forall \alpha \; !\!(\alpha \to \alpha) \to (\!\!(\alpha \to \alpha) \to (\alpha \to \alpha)) \]

\[ S \in \{0, 1\}^* \iff \text{proof } t_S \text{ of } \vdash \text{bint} \]
Aside on linear logic

\[ \pi \]

\[ !A, B \vdash C \]
Binary integers

\[
\text{bint} = \forall \alpha \left( !(\alpha \rightarrow \alpha) \rightarrow \left( !(\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha) \right) \right)
\]

\[S \in \{0, 1\}^* \quad \text{proof } t_S \text{ of } \vdash \text{bint}\]
Binary integers

\[ \text{bint} = \forall \alpha \, !(\alpha \rightarrow \alpha) \rightarrow !(\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha) \]

\( S \in \{0, 1\}^* \iff \text{proof } t_S \text{ of } \vdash \text{bint} \)

input \( g \)

input \( f \)

output \( f \circ f \circ f = g \circ f \circ f \)

001 \( \mapsto \{(f, g) \mapsto ff g\} \)

101 \( \mapsto \{(f, g) \mapsto gfg\} \)
Stratified Linear logic

variables: $\alpha, \beta, \gamma, \ldots$

formulas: $!F, \mathsection F, F \otimes F, F \multimap F, F \& F, \forall \alpha F$, constants

\[
\text{bint}^\mathsection = \forall \alpha !(\alpha \multimap \alpha) \multimap ((!(\alpha \multimap \alpha) \multimap \mathsection (\alpha \multimap \alpha))
\]

\[
\text{int}^\mathsection = \forall \alpha !(\alpha \multimap \alpha) \multimap \mathsection (\alpha \multimap \alpha)
\]
**Deduction rules for stratified linear logic**

same rules as before... e.g.

(Axiom): \[ \frac{}{A \vdash A} \]  

(Cut): \[ \frac{\Delta \vdash \Gamma \vdash A, \Delta', A \vdash B}{\Delta', \Gamma, \Delta \vdash B} \]

(Right $\to$): \[ \frac{A, \Gamma \vdash B}{\Gamma \vdash A \to B} \]

(Left $\to$): \[ \frac{\Gamma \vdash A}{\Delta', \Gamma, A \to B, \Delta \vdash C} \]

(Promotion): \[ \frac{!\Gamma \vdash A}{\Gamma \vdash !A} \]  

(Dereliction): \[ \frac{\Gamma, A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B} \]

(Contraction): \[ \frac{\Gamma, !A, !A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B} \]

(Weakening): \[ \frac{\Gamma, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B} \]

plus

\[ \frac{\Gamma, A \vdash B}{\Gamma, \text{prom } A \vdash B} \]

\[ \frac{\Gamma, \text{prom } A \vdash B}{\Gamma, A \vdash \text{prom } B} \]

\[ \frac{\Gamma, A \vdash B}{\Gamma, \text{prom } A \vdash B} \]

A proof in the stratified sequent calculus is a proof in the usual sense, together with a *stratification*, which is an assignment of integers to all occurrences of formulas, such that conclusions are assigned 0 and the assignment changes across deduction rules are as shown in blue.
Binary integers

\[
bint = \forall \alpha \, ! (\alpha \to \alpha) \to \left( ! (\alpha \to \alpha) \to (\alpha \to \alpha) \right)
\]

\[S \in \{0, 1\}^* \iff \text{proof } t_S \text{ of } \vdash \text{bint}\]
Binary integers (stratified)

\[
\text{bint}^\S = \forall \alpha \; !(\alpha \rightarrow \alpha) \rightarrow ( !(\alpha \rightarrow \alpha) \rightarrow \S(\alpha \rightarrow \alpha))
\]

\[S \in \{0, 1\}^* \quad \iff \quad \text{proof } t_S^\S \text{ of } \vdash \text{bint}^\S\]
Binary integers (stratified)

\[ \text{bint}^\$ = \forall \alpha ! (\alpha \rightarrow \alpha) \rightarrow ( !(\alpha \rightarrow \alpha) \rightarrow \$ (\alpha \rightarrow \alpha)) \]

\[ S \in \{0, 1\}^* \quad \text{proof } t^\$_S \text{ of } \vdash \text{bint}^\$ \]

\[ \begin{array}{c}
\vdash \text{bint}^\$ \\
\text{der, } \$ \\
\text{ctr} \\
\end{array} \]
Integers

\[ \text{int} = \forall \alpha \neg (\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha) \]

\( \forall n \in \mathbb{N} \) there is a proof \( n \) of \( \vdash \text{int} \)

Addition is a proof of \( \text{int} \), \( \text{int} \vdash \text{int} \)

Multiplication is a proof of \( \text{int} \), \( \text{int} \vdash \text{int} \)

A polynomial of degree \( k \) is a proof of \( \text{int} \vdash \text{int} \)
**Integers** (stratified)

\[
\text{int}^\S = \forall \alpha ! (\alpha \rightarrow \alpha) \rightarrow \S (\alpha \rightarrow \alpha)
\]

\(\forall n \in \mathbb{N} \) there is a proof \(n^\S\) of \(\vdash \text{int}^\S\)

(note that \(! (\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha)\) is not provable)

Addition is a proof of \(\text{int}^\S, \text{int}^\S \vdash \text{int}^\S\)

Multiplication is a proof of \(\text{int}^\S, \text{int}^\S \vdash \S \text{int}^\S\)

A polynomial of degree \(k\) is a proof of \(\text{int}^\S \vdash \S^k \text{int}^\S\)
Turing machines as proofs

Configuration \((w_L, w_R, q)\) of Turing machine \(\vdash \text{Tur}\)

Instructions for a Turing machine \(T\) \(\vdash \text{Tur} \rightarrow \text{Tur}\)
Running a Turing machine

(Identify a Turing machine $T$ with a proof of $\vdash \text{Tur} \rightarrow \text{Tur}$)

$T$

prepare initial state

\[
\vdash \text{Tur} \rightarrow \text{Tur} \quad \text{prom}
\]

\[
\vdash !(\text{Tur} \rightarrow \text{Tur})
\]

\[
\vdash \text{bint}, !(\text{Tur} \rightarrow \text{Tur}) \rightarrow (\text{Tur} \rightarrow \text{Tur}) \vdash \text{Tur}
\]

\[
\vdash \text{bint, int} \vdash \text{Tur}
\]

input binary integer

number of steps $n$ to run

config of Turing machine after $n$ steps
Running a Turing machine (stratified)

\[ \text{Tur}^\$ = \text{bint}^\$ \otimes \text{bint}^\$ \otimes \text{bool}_q^\$ \]

prepare initial state

\[
\begin{align*}
  & \vdash \text{Tur}^\$ \rightarrow \text{Tur}^\$ \\
  & \vdash \neg(\text{Tur}^\$ \rightarrow \text{Tur}^\$) \\
  & \vdash \text{bint}^\$, \text{Tur}^\$ \rightarrow \text{Tur}^\$ \vdash \text{Tur}^\$ \\
  & \vdash \text{bint}^\$, \neg(\text{Tur}^\$ \rightarrow \text{Tur}^\$) \vdash \text{Tur}^\$ \\
  & \vdash \text{bint}^\$, \text{int}^\$ \vdash \text{Tur}^\$
\end{align*}
\]
**Theorem** (Girard)

A function \( \{0, 1\}^* \rightarrow \{0, 1\}^* \) is “polytime” if and only if it can be typed as a proof \( \pi \) of \( \text{bint} \vdash \text{bint} \) which admits a stratification.

\[
\begin{align*}
\pi^{\text{§}} & \quad \vdash \quad \pi \\
\vdash \quad \text{stratifies} \quad \vdash \\
\text{bint}^{\text{§}} & \vdash \quad \text{bint}^{\text{§}} \\
& \quad \vdash \quad \text{bint} \vdash \text{bint}
\end{align*}
\]
Theorem (Girard)

A function \( \{0, 1\}^* \rightarrow \{0, 1\}^* \) is “polytime” if and only if it can be typed as a proof \( \pi \) of \( \text{bint} \vdash \text{bint} \) which admits a stratification.

\[
f : \{0, 1\}^* \rightarrow \{0, 1\}^* \text{ computed by a Turing machine } T \text{ with polyclock } P
\]
Theorem (Girard)

A function \( \{0, 1\}^* \rightarrow \{0, 1\}^* \) is “polytime” if and only if it can be typed as a proof \( \pi \) of \( \text{bint} \vdash \text{bint} \) which admits a stratification.

\[ f : \{0, 1\}^* \rightarrow \{0, 1\}^* \text{ computed by a Turing machine } T \text{ with polyclock } P \]

Upshot: \( \pi \) computes \( f \)
Theorem (Girard)

A function \( \{0, 1\}^* \rightarrow \{0, 1\}^* \) is "polytime" if and only if it can be typed as a proof \( \pi \) of \( \text{bint} \vdash \text{bint} \) which admits a stratification.

\[ f : \{0, 1\}^* \rightarrow \{0, 1\}^* \text{ computed by a Turing machine } T \text{ with polyclock } P \]

\[
\begin{array}{ccc}
\text{copy} & \text{P} & \text{iterate } T \\
\vdots & \vdots & \vdots \\
\text{bint}^\Sigma \vdash \Sigma(bint^\Sigma \otimes \text{bint}^\Sigma) & \text{int}^\Sigma \vdash \Sigma^k \text{int}^\Sigma & \Sigma\text{bint}^\Sigma, \text{int}^\Sigma \vdash \Sigma\text{Tur}^\Sigma \\
\end{array}
\]

\[
\begin{array}{ccc}
\pi^\Sigma & \text{stratifies} & \pi \\
\vdots & \vdots & \\
\text{bint}^\Sigma \vdash \Sigma^{k+2} \text{bint}^\Sigma & \text{bint} \vdash \text{bint} \\
\end{array}
\]
Summary

• There is a notion of *stratification* for proofs

• Turing machines can be encoded into linear logic

• If a Turing machine is polytime, the stratification of the clock polynomial gives a stratification of the corresponding proof in linear logic.

• Theorem: a function of binary integers is polytime iff. it admits a stratification.
References


Slides of this lecture available at therisingsea.org