Logical Methods in NLP

Intro Part 2

Michael Moortgat

Contents

1. Substructural logics

- Intuitionistic logic
- Linear logic
- Lambek logics

Logical vs structural rules

In presenting the rules for valid reasoning, one finds two types of rules:

- logical rules: tell you how to use and derive complex propositions out of simpler ones
- structural rules: tell you how you can manipulate assumptions in constructing a proof. For example: by permuting, copying or deleting them

Sub structural logics result from dropping some/all of the structural rules that traditional logic adopts.

- no copying/deletion: assumptions become resources, used up in reasoning
- no permutation: the linear order of resources matters

This leads to logics that are more appropriate for our concerns: logics of perception, action, natural language syntax and semantics (but also: economics, games, quantum mechanics . . .)

Standard logic

Formulas Let's just look at conjunction and implication . . .

$$A, B ::= p \mid A \times B \mid A \to B$$

- p : atomic propositions
- $A \times B$: conjunction, 'A and B'
- $A \to B$: implication, 'if A then B'

Judgements In an intuitionistic world: multiple assumptions, single conclusion.

$$\Gamma \vdash A$$

- from assumptions Γ one can conclude proposition A.
- we write Γ for a sequence of zero or more propositions.

Standard logic: natural deduction rules

$$\frac{}{A \vdash A}\ \text{Ax} \qquad \frac{\Gamma, \Delta \vdash A}{\Delta, \Gamma \vdash A}\ \text{Exchange}$$

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\ \text{Contraction} \qquad \frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ \text{Weakening}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \to I \qquad \frac{\Gamma \vdash A \to B \quad \Delta \vdash A}{\Gamma, \Delta \vdash B} \to E$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \times B} \times I \qquad \frac{\Gamma \vdash A \times B \quad \Delta, A, B \vdash C}{\Gamma, \Delta \vdash C} \times E$$

- Exchange: order of assumptions doesn't matter
- Contraction: assumptions are re-usable
- Weakening: assumptions can be thrown away

Another conjunction?

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \times B} \times I' \qquad \frac{\Gamma \vdash A \times B}{\Gamma \vdash A} \times E_1' \qquad \frac{\Gamma \vdash A \times B}{\Gamma \vdash B} \times E_2'$$

Not really . . . In the presence of Contraction, Weakening, (and Exchange) the different formulations are interderivable. For example, obtaining $(\times')$ from $(\times I)$

$$\cfrac{\cfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma, \Gamma \vdash A \times B} \times I}{\Gamma \vdash A \times B} \text{ Contraction}$$

This shows that the structural rules blur the picture of what logical constants one can distinguish.

Linear logic: logic of resources

- Contraction, Weakening are no longer freely available
- assumptions become finite, material resources

Formulas

$$A, B ::= p \mid A \otimes B \mid A \,\&\, B \mid A \multimap B \mid \ldots$$

- p : atomic propositions
- $A \otimes B$: multiplicative conjunction, 'both A and B'
- $A \,\&\, B$: additive conjunction, 'choose from A and B'
- $A \multimap B$: linear implication, 'consume A producing B'

Linear logic: natural deduction rules

$$\frac{}{A \vdash A}\ \mathsf{Ax} \qquad \frac{\Gamma, \Delta \vdash A}{\Delta, \Gamma \vdash A}\ \mathsf{Exchange}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \multimap I \qquad \frac{\Gamma \vdash A \multimap B \quad \Delta \vdash A}{\Gamma, \Delta \vdash B} \multimap E$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \otimes I \qquad \frac{\Gamma \vdash A \otimes B \quad \Delta, A, B \vdash C}{\Gamma, \Delta \vdash C} \otimes E$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B} \,\&I \qquad \frac{\Gamma \vdash A \,\&\, B}{\Gamma \vdash A} \,\&E_1 \qquad \frac{\Gamma \vdash A \,\&\, B}{\Gamma \vdash B} \,\&E_2$$

Universal Logic

Linear and intuitionistic constants can coexist in a 'universal logic' (Girard 1991).

- assumptions are sorted as linear: $\langle A \rangle$ or intuitionistic: $[A]$
- Contraction/Weakening come back in a controlled form: $!A$ ('of course A')

On the next page, $\oplus$ stands for additive disjunction (dual to $\&$)

Communication The split conjunctions (multiplicative, additive) communicate via !

$$\langle\, !(A \,\&\, B)\,\rangle \vdash\, !\,A \otimes !\,B \qquad \langle\, !\,A \otimes !\,B\,\rangle \vdash\, !(A \,\&\, B)$$

EXERCISE prove these equivalences.

$$\frac{}{\langle A\rangle \vdash A}\,\langle\mathrm{Id}\rangle \qquad \frac{}{[A] \vdash A}\,[\mathrm{Id}] \qquad \frac{\Gamma,\ \Delta \vdash A}{\Delta,\ \Gamma \vdash A}\,\text{Exchange}$$

$$\frac{\Gamma,\ [A],\ [A] \vdash B}{\Gamma,\ [A] \vdash B}\,\text{Contraction} \qquad \frac{\Gamma \vdash B}{\Gamma,\ [A] \vdash B}\,\text{Weakening}$$

$$\frac{[\Gamma] \vdash A}{[\Gamma] \vdash\ !A}\,!\text{-I} \qquad \frac{\Gamma \vdash\ !A \qquad \Delta,\ [A] \vdash B}{\Gamma,\ \Delta \vdash B}\,!\text{-E}$$

$$\frac{\Gamma,\ \langle A\rangle \vdash B}{\Gamma \vdash A \multimap B}\,\multimap\text{-I} \qquad \frac{\Gamma \vdash A \multimap B \qquad \Delta \vdash A}{\Gamma,\ \Delta \vdash B}\,\multimap\text{-E}$$

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma,\ \Delta \vdash A \otimes B}\,\otimes\text{-I} \qquad \frac{\Gamma \vdash A \otimes B \qquad \Delta,\ \langle A\rangle,\ \langle B\rangle \vdash C}{\Gamma,\ \Delta \vdash C}\,\otimes\text{-E}$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}\,\&\text{-I} \qquad \frac{\Gamma \vdash A \,\&\, B}{\Gamma \vdash A}\,\&\text{-E}_1 \qquad \frac{\Gamma \vdash A \,\&\, B}{\Gamma \vdash B}\,\&\text{-E}_2$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\,\oplus\text{-I}_1 \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\,\oplus\text{-I}_2 \qquad \frac{\Gamma \vdash A \oplus B \qquad \Delta,\ \langle A\rangle \vdash C \qquad \Delta,\ \langle B\rangle \vdash C}{\Gamma,\ \Delta \vdash C}\,\oplus\text{-E}$$

Embedding

Linear logic has a more finegrained view on the logical constants than intuitionistic logic. But thanks to !, no expressivity is lost:

$$\Gamma \vdash A \text{ is provable intuitionistically iff } [\Gamma] \vdash A \text{ is provable in linear logic}$$

Embedding translation

$$\begin{array}{rcl} A \to B & = & !\,A \multimap B \\ A \times B & = & A \,\&\, B \\ A + B & = & !\,A \oplus !\,B \end{array}$$

EXERCISE Show that the intuitionistic rules for $\times$, $+$ can be derived from the corresponding rules of linear logic, together with the ! intro/elim rules.

Lambek calculus: logic of structured resources

Why stop here? By removing the remaining structural rules, one obtains logics of structured grammatical resources.

- dropping Exchange: logic of strings, word order sensitivity (**L**)
- dropping rebracketing: logic of phrases, constituent structure (**NL**)

Lambek 1958, 1961 (compare linear logic: Girard 1987)

Formulas

$$A, B ::= p \mid A \otimes B \mid A \backslash B \mid B/A \mid \ldots$$

- $A \otimes B$: composition, 'A and then B'
- $A \backslash B$: left imcompleteness, 'consume A to the left producing B'
- B/A: right incompleteness, 'consume A to the right producing B'

Lambek calculus: strings of assumptions

$$\frac{}{A \vdash A}\ \textsf{Ax}$$

$$\frac{A, \Gamma \vdash B}{\Gamma \vdash A\backslash B}\ \backslash I \qquad \frac{\Gamma \vdash A \quad \Delta \vdash A\backslash B}{\Gamma, \Delta \vdash B}\ \backslash E$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash B/A}\ /I \qquad \frac{\Gamma \vdash B/A \quad \Delta \vdash A}{\Gamma, \Delta \vdash B}\ /E$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ \otimes I \qquad \frac{\Gamma \vdash A \otimes B \quad \Delta, A, B, \Delta' \vdash C}{\Delta, \Gamma, \Delta' \vdash C}\ \otimes E$$

Associativity The comma still hides a structural rule. Structures $\Gamma := A \mid A, \Gamma$

Lambek calculus: bracketed strings of assumptions

2-place structure-building operation $\circ$: structures $S := A \mid (S \circ S)$

Notation: $\Gamma[\Delta]$ structure Γ with substructure Δ (see $\otimes$ Elimination)

$$\frac{}{A \vdash A}\ \mathsf{Ax}$$

$$\frac{(A \circ \Gamma) \vdash B}{\Gamma \vdash A \backslash B}\ \backslash I \qquad \frac{\Gamma \vdash A \quad \Delta \vdash A \backslash B}{(\Gamma \circ \Delta) \vdash B}\ \backslash E$$

$$\frac{(\Gamma \circ A) \vdash B}{\Gamma \vdash B/A}\ /I \qquad \frac{\Gamma \vdash B/A \quad \Delta \vdash A}{(\Gamma \circ \Delta) \vdash B}\ /E$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{(\Gamma \circ \Delta) \vdash A \otimes B}\ \otimes I \qquad \frac{\Gamma \vdash A \otimes B \quad \Delta[(A \circ B)] \vdash C}{\Delta[\Gamma] \vdash C}\ \otimes E$$

Explicit structural rules

Instead of using the hypocritical comma, one can explicitly introduce the structural rules of Exchange, Associativity.

Non-logical axioms versus rules Replace formula variables by structure variables.

$$A \otimes B \vdash B \otimes A \qquad \frac{X[(Z \circ Y)] \vdash C}{X[(Y \circ Z)] \vdash C}\ \text{Exchange}$$

$$A \otimes (B \otimes C) \dashv\vdash (A \otimes B) \otimes C \qquad \frac{X[(Y \circ Z) \circ W] \vdash D}{X[Y \circ (Z \circ W)] \vdash D}\ \text{Assoc}$$

Structural control

We don't want to completely drop Exchange, Rebracketing, but bring them back in a controlled form. Compare: Contraction/Weakening in the intuitionistic/linear case.

Control operators A pair of unary type-forming operations:

$$A, B \quad ::= \quad p \mid \Diamond A \mid \Box A \mid A \otimes B \mid A/B \mid B\backslash A$$

Structures Next to the 2-place structure-building operation $\cdot \circ \cdot$ now also 1-place $\langle\cdot\rangle$ as structural counterpart of $\Diamond$.

$$X, Y \quad ::= \quad A \mid \langle X \rangle \mid X \circ Y$$

Logical rules $\Diamond, \Box$ form a residuated pair: $\Diamond\Box A \vdash A \vdash \Box\Diamond A$.

$$\frac{X \vdash \Box A}{\langle X \rangle \vdash A}\ \Box E \qquad \frac{\langle X \rangle \vdash A}{X \vdash \Box A}\ \Box I$$

$$\frac{X \vdash A}{\langle X \rangle \vdash \Diamond A}\ \Diamond I \qquad \frac{Y \vdash \Diamond A \quad X[\langle A \rangle] \vdash B}{X[Y] \vdash B}\ \Diamond E$$

Embeddings

Structural control can be realized in two ways:

- $\diamondsuit$ as licence: allow a structural rule that would not be available without $\diamondsuit$
- $\diamondsuit$ as obstacle: block a structural option that would otherwise be possible

Structural control $\mathcal{L}$, $\mathcal{L}'$ two logics that differ w.r.t. structural option P: $\mathcal{L}' = \mathcal{L} + P$. We express $\mathcal{L}$ in $\mathcal{L}'$ or vv, via translations $\cdot^\flat$, $\cdot^\sharp$:

(obstacle) $A \vdash B$ is provable in $\mathcal{L}_{/,\otimes,\backslash}$ iff $A^\flat \vdash B^\flat$ is provable in $\mathcal{L}'_{\diamondsuit,\Box,/,\otimes,\backslash}$

the translation blocks applications of P

(licence) $A \vdash B$ is provable in $\mathcal{L}'_{/,\otimes,\backslash}$ iff $A^\sharp \vdash B^\sharp$ is provable in $\mathcal{L}_{\diamondsuit,\Box,/,\otimes,\backslash} + P_\diamond$

$P_\diamond$: image of P under $\cdot^\sharp$; allowing a 'modal' version of P

(Kurtonina & MM '97)

Illustration: NL versus L

Let $\mathcal{L}$ be the base logic **NL** (no structural rules at all) and $\mathcal{L}'$ the string logic **L**, with associative tensor.

Translations One schema fits both $\cdot^\flat$ and $\cdot^\sharp$

$$\begin{array}{rcl} p^\natural & = & p \\ (A \otimes B)^\natural & = & \Diamond(A^\natural \otimes B^\natural) \\ (A/B)^\natural & = & \Box A^\natural / B^\natural \\ (B\backslash A)^\natural & = & B^\natural \backslash \Box A^\natural \end{array}$$

$\Diamond$ as obstacle $\cdot^\flat$ expresses **NL** in **L**: $\Diamond$ blocks all possible applications of (A). Example: the $\cdot^\flat$ translation of (†) fails.

$$\dagger \quad (a\backslash b) \otimes (b\backslash c) \vdash a\backslash c$$

$\Diamond$ as licence With $\cdot^\sharp$, **L** can be expressed in **NL** $+ A_\diamond$. $\Diamond$ provides access to a modal version of (A). The $\cdot^\sharp$ translation of (†) is derivable.

$$\Diamond(\Diamond(A \otimes B) \otimes C) \dashv\vdash \Diamond(A \otimes \Diamond(B \otimes C)) \quad (A_\diamond) = (A)^\sharp$$

Expressivity, complexity

- **NL**: context-free; polynomial
- **L**: context-free; NP complete (with fixed lexicon: polynomial)
- **LP**: permutation-closures of CF languages; NP complete
- **NL**$\Diamond$: depends on restrictions on structural options
  - linear, non-expanding: context-sensitive; PSPACE (Moot 2002)
  - controlled extraction: mildly CS (TAG); polynomial (Moot 2008)

Controlled (left) extraction Cf MG 'Move'

$$\Diamond A \otimes (B \otimes C) \vdash (\Diamond A \otimes B) \otimes C$$

$$\Diamond A \otimes (B \otimes C) \vdash B \otimes (\Diamond A \otimes C)$$

Symmetrically for controlled rightward displacement.

2. Proofs and terms

For the 'meaning of proofs', we build on two central ideas.

- Montague: compositional interpretation as a structure-preserving mapping relating a source calculus to a target calculus. In the case of Lambek grammars, we can take **NL** and **LP** as source and target, speaking about composition in the form dimension and in the meaning dimension respectively.
- Curry: the Curry-Howard correspondence, linking systems of logical deduction (intuitionistic logic) and models of computation (the simply typed lambda calculus). In the case of Lambek grammars, we find a resource-conscious version of the correspondence. Derivations in the target calculus **LP** are associated with terms of the simply typed *linear* lambda calculus. These terms express instructions for meaning assembly, disallowing copying or deletion of grammatical material.

Compositionality

- central design principle of computational semantics: Frege's principle

  'the meaning of an expression is a function of the meaning of its parts and of the way they are syntactically combined' (Partee)

- Montague's Universal Grammar program: compositionality as a homomorphism

$$\langle (A_s)_{s\in S}, F\rangle \quad \xrightarrow{\quad h \quad} \quad \langle (B_t)_{t\in T}, G\rangle$$

  - source: algebra A with sorts (categories) S, operations F ('abstract syntax')
  - target: algebra B with sorts (types) T, operations G ('interpretation')
  - homomorphism h: a mapping that respects (i) sorts and (ii) operations:

$$(i) \quad h[A_s] \subseteq B_t \quad (t\text{: target sort corresponding to } s)$$

$$(ii) \quad h(f(a_1,\ldots,a_n)) \quad = \quad g(h(a_1),\ldots,h(a_n))$$

$$(g\text{: target operation corresponding to } f)$$

Lambek calculi and compositionality

$$\mathbf{(N)L}^{\{n,np,s\}}_{/,\backslash} \xrightarrow{\quad\quad h \quad\quad} \mathbf{LP}^{\{e,t\}}_{\to}$$

source homomorphism target

- source: syntactic calculus **NL**
  - atomic types: distinct kinds of phrases
  - operations: directional, prefixation/suffixation
- target: semantic calculus **LP**
  - atomic types: distinct kinds of semantic objects
  - operations: non-directional function types

To Be Done

- how does h act on types? mapping of source types to target types
- how does h act on derivations? mapping source N.D. proofs to target proofs

Curry-Howard Correspondence

Slogans

formulas-as-types

proofs-as-programs

Logic and computation deep connection between logical derivations and programs

INTUITIONISTIC LOGIC	LAMBDA CALCULUS
formulas	types
connectives	type constructors
implication	function space
proofs	terms
assumption	variable
normalization	reduction
⋮	⋮

Simple types, denotation domains, signatures

Simple types given a finite set of atomic types $\mathcal{A}$, we define $\mathcal{T}_{\mathcal{A}}$, the set of simple types constructed from $\mathcal{A}$, as follows:

$$\mathcal{T}_{\mathcal{A}} \quad ::= \quad \mathcal{A} \quad | \quad \mathcal{T}_{\mathcal{A}} \rightarrow \mathcal{T}_{\mathcal{A}}$$

Denotation domains For every $A \in \mathcal{T}_{\mathcal{A}}$, we provide a semantic domain D_A where terms of type A find their possible interpretation. After stipulating domains for the atomic types, we set

$$D_{A \rightarrow B} \quad = \quad D_B^{D_A} \qquad \text{the set of functions from } D_A \text{ to } D_B$$

Signature type assignment to constants: $\Sigma = \langle \mathcal{A}, C, \tau \rangle$

- $\mathcal{A}$: finite set of atomic types
- C : finite set of constants
- $\tau : C \longrightarrow \mathcal{T}_{\mathcal{A}}$ type assignment function

Terms, typing rules

Terms Given set of variables $\mathcal{X}$ and signature $\Sigma = \langle \mathcal{A}, C, \tau \rangle$, the set of lambda terms built upon Σ is inductively defined as ($x \in \mathcal{X}$, $c \in C$)

$$T \quad ::= \quad x \quad | \quad c \quad | \quad \lambda x.T \quad | \quad (T\ T)$$

Typing rules Sequents as typing judgements: we read $\Gamma \vdash t : \alpha$ as: term t can be assigned type α given Γ, a set of type declarations $x_i : A_i$ (a 'typing environment')

$$\Gamma, x : \alpha \vdash x : \alpha \quad (var) \qquad \Gamma \vdash c : \tau(c) \quad (cons)$$

$$\frac{\Gamma, x : \alpha \vdash t : \beta}{\Gamma \vdash \lambda x.t : (\alpha \to \beta)}\ (abs)$$

$$\frac{\Gamma \vdash t : (\alpha \to \beta) \quad \Gamma \vdash u : \alpha}{\Gamma \vdash (t\ u) : \beta}\ (app)$$

Note Γ copied in (app), unused in $(var), (cons)$. (app): $\to$ Elim, (abs): $\to$ Intro.

Illustration

Basic types Let $\mathcal{A}$ be $\{n, t\}$, with

$$D_n = \mathbb{N} \quad \text{(the natural numbers } \{0, 1, 2, \ldots\}) \quad ; \quad D_t = \{\mathbf{t}, \mathbf{f}\} \quad \text{(boolean values)}$$

Signature assume we have some constants with the types below:

	τ
times, plus	$n \to n \to n$
leq	$n \to n \to t$
odd, even	$n \to t$

Typing a term let's show that the program $\lambda x.(\text{times } x\ x)$ is of type $n \to n$

$$\dfrac{\dfrac{}{x : n \vdash x : n}\ var \qquad \dfrac{\dfrac{}{x : n \vdash x : n}\ var \qquad \dfrac{}{x : n \vdash \text{times} : n \to n \to n}\ cons}{x : n \vdash (\text{times } x) : n \to n}\ app}{\dfrac{x : n \vdash (\text{times } x\ x) : n}{\vdash \lambda x.(\text{times } x\ x) : n \to n}\ abs}\ app$$

Linear lambda calculus

In Curry's original set-up for IL, λ can bind multiple occurrences of a parameter, or bind a variable that doesn't occur in the body of the abstraction. For natural language computations, we want a more restricted regime.

- Intuitionistic logic (IL): copying, deletion of assumptions freely available
- Linear Logic (LL): assumptions as resources; every assumption used exactly once

Linear typing rules in judgements $\Gamma \vdash t : \alpha$, the environment Γ is now a multiset

$$x : \alpha \vdash x : \alpha \quad (var) \qquad \vdash c : \tau(c) \quad (cons)$$

$$\frac{\Gamma, x : \alpha \vdash t : \beta}{\Gamma \vdash \lambda x.t : (\alpha \to \beta)} \quad x \notin \mathrm{dom}(\Gamma) \qquad (abs)$$

$$\frac{\Gamma \vdash t : (\alpha \to \beta) \quad \Delta \vdash u : \alpha}{\Gamma, \Delta \vdash (t\ u) : \beta} \quad \mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Delta) = \emptyset \qquad (app)$$

Remark We'll allow non-linearity for lexical (vs derivational) semantics.

Proof normalisation and term reduction

Removal of detours form N.D. proofs $\rightsquigarrow$ simplification of terms.

On the left: redex; on the right: contractum. Same interpretation under $[\![\cdot]\!]_I^g$.

η reduction compare $\to$ Elimination (app) immediately followed by Intro (abs).

$$\dfrac{\dfrac{x:\alpha,\Gamma \vdash (t\ x) : \beta}{\Gamma \vdash \lambda x.(t\ x) : \alpha \to \beta}\,abs}{}\quad\text{where}\quad \dfrac{\dfrac{}{x:\alpha \vdash x:\alpha}\,var \qquad \dfrac{\vdots}{\Gamma \vdash t : \alpha \to \beta}}{x:\alpha,\Gamma \vdash (t\ x):\beta}\,app \qquad \rightsquigarrow_\eta \qquad \dfrac{\vdots}{\Gamma \vdash t : \alpha \to \beta}$$

β reduction compare $\to$ Intro (abs) immediately followed by Elimation (app)

$$\dfrac{\dfrac{\vdots}{\Delta \vdash u : \alpha} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{x:\alpha \vdash x:\alpha}\,var}{\vdots}}{x:\alpha,\Gamma \vdash t:\beta}}{\Gamma \vdash \lambda x.t : \alpha \to \beta}\,abs}{\Delta,\Gamma \vdash ((\lambda x.t)\ u) : \beta}\,app \qquad \rightsquigarrow_\beta \qquad \dfrac{\dfrac{\dfrac{\vdots}{\Delta \vdash u:\alpha}}{\vdots}}{\Delta,\Gamma \vdash t[x \mapsto u] : \beta}$$

Proofs and terms: the complete picture

We concentrated so far on the implicational fragment (enough for ACG). Curry-Howard for a fuller formula language ($\multimap, !, \otimes, \&, \oplus$) see Wadler.

$$
\begin{array}{rl}
s, t, u, v, w ::= & x \\
| & \lambda\langle x\rangle.\, u \mid s\,\langle t\rangle \\
| & !t \mid \text{case } s \text{ of } !x \to u \\
| & \langle t, u\rangle \mid \text{case } s \text{ of } \langle x, y\rangle \to v \\
| & \langle\!\langle t, u\rangle\!\rangle \mid \text{fst}\,\langle s\rangle \mid \text{snd}\,\langle s\rangle \\
| & \text{inl}\,\langle t\rangle \mid \text{inr}\,\langle u\rangle \mid \text{case } s \text{ of inl}\,\langle x\rangle \to v;\ \text{inr}\,\langle y\rangle \to w
\end{array}
$$

Proof reductions — term reductions:

$$
\begin{array}{rl}
\text{case } !t \text{ of } !x \to u & \Longrightarrow u[t/x] \\
(\lambda\langle x\rangle.\, u)\,\langle t\rangle & \Longrightarrow u[t/x] \\
\text{case } \langle t, u\rangle \text{ of } \langle x, y\rangle \to v & \Longrightarrow v[t/x, u/y] \\
\text{fst}\,\langle\, \langle\!\langle t, u\rangle\!\rangle\,\rangle & \Longrightarrow t \\
\text{snd}\,\langle\, \langle\!\langle t, u\rangle\!\rangle\,\rangle & \Longrightarrow u \\
\text{case inl}\,\langle t\rangle \text{ of inl}\,\langle x\rangle \to v;\ \text{inr}\,\langle y\rangle \to w & \Longrightarrow v[t/x] \\
\text{case inr}\,\langle u\rangle \text{ of inl}\,\langle x\rangle \to v;\ \text{inr}\,\langle y\rangle \to w & \Longrightarrow w[u/y]
\end{array}
$$

Directional types, terms

Syntactic source calculus linear implication $\to$ splits in two directional implications.

Directional types given a finite set of atomic types $\mathcal{A}$, and $p \in \mathcal{A}$

$$A, B \quad ::= \quad p \quad | \quad A \backslash B \quad | \quad B / A$$

Directional terms given a set of variables $\mathcal{X}$,

$$M, N \quad ::= \quad x \quad | \quad \lambda^r x.M \quad | \quad \lambda^l x.M \quad | \quad (M \triangleleft N) \quad | \quad (N \triangleright M)$$

Directional typing rules the terms record the distinction between $\backslash$ and $/$

$$\frac{\Gamma \circ x : A \vdash M : B}{\Gamma \vdash \lambda^r x.M : B/A}\ I/ \qquad \frac{x : A \circ \Gamma \vdash M : B}{\Gamma \vdash \lambda^l x.M : A\backslash B}\ I\backslash$$

$$\frac{\Gamma \vdash M : B/A \quad \Delta \vdash N : A}{\Gamma \circ \Delta \vdash (M \triangleleft N) : B}\ E/ \qquad \frac{\Gamma \vdash N : A \quad \Delta \vdash M : A\backslash B}{\Gamma \circ \Delta \vdash (N \triangleright M) : B}\ E\backslash$$

Compositional interpretation

$$\textbf{(N)L}^{\{n,np,s\}}_{/,\backslash} \quad \xrightarrow{\quad (\cdot)' \quad} \quad \textbf{LP}^{\{e,t\}}_{\rightarrow}$$

Types domains $D_e = E$ (entities, individuals), $D_t = \{0,1\}$ (truth values)

$$np' = e \quad s' = t \quad n' = e \to t \quad (A\backslash B)' = (B/A)' = A' \to B'$$

Terms We write $\widetilde{x}$ for the target variable corresponding to source variable x

$$\begin{array}{rcl} x' & = & \widetilde{x} \\ (\lambda^l x.M)' = (\lambda^r x.M)' & = & \lambda\widetilde{x}.M' \\ (N \triangleright M)' = (M \triangleleft N)' & = & (M'\ N') \end{array}$$

Example

Source **NL** sequent $\Gamma \vdash t : B$ where Γ is a tree with leafs $x_i : A_i$ and t a directional linear lambda term of type B built from the $x_i : A_i$. We call the x_i the syntactic parameters of the derivation; t the proof term.

$$\dfrac{\dfrac{x}{x : s/(np\backslash s)} \quad \dfrac{\dfrac{y}{y : (np\backslash s)/np} \quad \dfrac{z}{z : ((np\backslash s)/np)\backslash(np\backslash s)}}{y \circ z \vdash (y \triangleright z) : np\backslash s}[\backslash E]}{x \circ (y \circ z) \vdash (x \triangleleft (y \triangleright z)) : s}[/E]$$

Target **LP** sequent $\widetilde{\Gamma} \vdash t' : B'$; $\widetilde{\Gamma}$ a multiset of assumptions $\widetilde{x}_i : A'_i$ and t' a linear lambda term built from the semantic parameters $\widetilde{x}_i$.

$$\dfrac{\dfrac{\widetilde{x}}{\widetilde{x} : (e \to t) \to t} \quad \dfrac{\dfrac{\widetilde{y}}{\widetilde{y} : e \to e \to t} \quad \dfrac{\widetilde{z}}{\widetilde{z} : (e \to e \to t) \to e \to t}}{\widetilde{y}, \widetilde{z} \vdash (\widetilde{z}\ \widetilde{y}) : e \to t}[\to E]}{\widetilde{x}, \widetilde{y}, \widetilde{z} \vdash (\widetilde{x}\ (\widetilde{z}\ \widetilde{y})) : t}[\to E]$$

Next step substitute $x_i \mapsto$ word$_i$; $\widetilde{x}_i \mapsto$ terms expressing the lex semantics of word$_i$

Lost in translation

Desirable semantic terms are often unobtainable as image of **(N)L** proofs:

$$(\Lambda_{\mathbf{NL}})' \subset (\Lambda_{\mathbf{L}})' \subset \Lambda_{\mathbf{LP}}$$

Recovering lost expressivity Two strategies:

- **(N)L**$\diamondsuit$ source with controlled structural rules
- ACG: **LP** source; surface form itself obtained via interpretation mapping

3. Proof nets

(Non-)efficiency Sequent proof search has two types of non-determinism in the choice of the active formula:

- Don't know: different choices lead to logically non-equivalent interpretations ('readings')
- Don't care: different choices lead to one and the same reading

For completeness, the first form of non-determinism has to be kept;

for efficiency, the second form has to be eliminated.

$\rightsquigarrow$ proof nets!

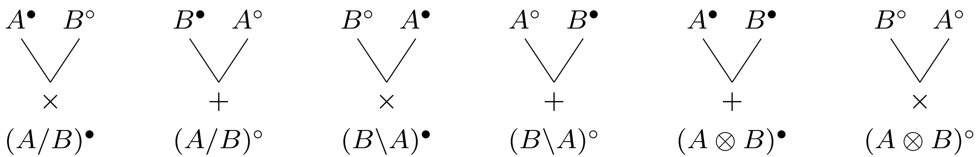
Compare dependency structures induced by normal derivations.

Polarised formulas

In the intuitionistic (single conclusion) world, we consider formulas with polarities:

- $\cdot^{\bullet}$ polarity: input, antecedent, 'given'
- $\cdot^{\circ}$ polarity: output, succedent, 'to prove'

Formula decomposition With the following unfolding rules, we compute the formula decomposition tree for arbitrary (input/output) formulas.



$\times$-links versus $+$-links

The formula decomposition rules distinguish two types of links:

- $\times$-type ('tensor') links: cf. the two-premise sequent rules $/L$, $\backslash L$, $\otimes R$
- $+$-type ('cotensor'/'par') links: cf. the one-premise sequent rules $\otimes L$, $/R$, $\backslash R$

The order of the subtypes in the premises is significant; it is inverted in the $\cdot^{\circ}$ unfolding.

An invariant for L: well-bracketing

For x a list of polarized formulas, let $\ell(x)$ be the yield of the formula decomposition trees given by the mappings $\cdot^\bullet$, $\cdot^\circ$.

Example $\ell((a/b)^\bullet((c/a)\backslash(c/b))^\circ) = a^\bullet b^\circ b^\bullet c^\circ c^\bullet a^\circ$.

Let r, s, t, u, v, w range over lists of polarized atomic formulas. Let $\pm, \mp$ range over opposite polarities.

Definition r is well-bracketed if $r = \epsilon$ or $r = p^\pm s p^\mp t$, where s, t are well-bracketed.

Theorem If $\Gamma \Rightarrow A$ is **(N)L** provable, then $\ell(\Gamma^\bullet A^\circ)$ is well-bracketed.

Proof Induction on the sequent derivation, using the fact that (i) if s, t are well-bracketed, then also st; (ii) if st, u are well-bracketed, then also sut; (iii) if st is well-bracketed, then also ts.

Proof nets: inductive definition

Let β, γ range over proof nets, t, u over lists of polarized formulas; A, B over polarized formulas; $*$ over connectives.

axiom $\overline{a^\bullet \quad a^\circ}$ and $\overline{a^\circ \quad a^\bullet}$ are proof nets with terminal formulas (tf) $a^\bullet a^\circ$, $a^\circ a^\bullet$;

+-link if β is a proof net with tf $tABu$, we obtain a new net with tf $tC * Du$ by applying $\dfrac{A \quad B}{C * D}\, +$;

×-link if β, γ are nets with tf tA, uBv, we obtain a new net

- with tf $utC * Dv$, by applying $\dfrac{A \quad B}{C * D}\, \times$;
- with tf $uC * Dtv$, by applying $\dfrac{B \quad A}{C * D}\, \times$;

cperm if β is a net with tf t, we can apply any cyclic permutation to t provided we preserve the linkage.

Soundness, completeness

Definition A proof net β is planar if $\ell(\beta)$ is well-bracketed, i.e. its axiom links do not cross.

Completeness Every **(N)L** sequent derivation π can be transformed into a planar proof net $\beta(\pi)$.

Soundness Every planar proof net β can be translated back to an **L** sequent derivation.

(Proofs omitted)

Remark For **NL** soundness, the well-bracketing invariant (planarity) is not enough. See below for the extra condition (operator balance).

Proof structures, proof nets

To build a proof net for an **(N)L(P)** sequent $\Gamma \Rightarrow B$, where Γ is a structure with yield $A_1, \ldots, A_n$, proceed as follows:

1. Build a candidate proof structure. For **L(P)** this is the list of formula decomposition trees $A_1^\bullet \ldots A_n^\bullet B^\circ$ together with an axiom linking. In the case of **NL**, the antecedent $\circ$ structure is translated in $(- \otimes -)^\bullet$ links.
2. Check whether the proof structure is in fact a proof net by testing the relevant correctness criteria.
   - **LP**. A correction graph is obtained from a proof structure by removing exactly one edge from every $+$-link. A proof structure is a proof net for **LP** iff every correction graph for it is a-cyclic and connected
   - **L**. The proof structure has a planar axiom linking.
   - **NL**. Operator balance: every cycle of the proof structure has an equal number of $\times$ and $+$.

Remark In the case of (associative) **L**, we don't bother to impose an explicit structure on the formulas in Γ: the sequent antecedent is simply treated as a list of formulas.

Nets and their lambda terms

We compute the Curry-Howard lambda term for a proof net. Below the rules for the $/, \backslash$ fragments.

Notation $x, y, \ldots$ for object-level variables; M, N for meta-level variables; $t, u, \ldots$ for terms built out of object and meta variables.

$$\frac{(t\ M) : A^{\bullet} \quad M : B^{\circ}}{t : A/B^{\bullet}} \times \qquad \frac{x : B^{\bullet} \quad N : A^{\circ}}{\lambda x.N : A/B^{\circ}} +$$

$$\frac{M : B^{\circ} \quad (t\ M) : A^{\bullet}}{t : B\backslash A^{\bullet}} \times \qquad \frac{N : A^{\circ} \quad x : B^{\bullet}}{\lambda x.N : B\backslash A^{\circ}} +$$

Axiom links (One-sided) unification/matching of the unknown M at the output node with the term t at the input node.

$$\frac{\{M := t\}}{t : p^{\bullet} \qquad M : p^{\circ}} \qquad \frac{\{M := t\}}{M : p^{\circ} \qquad t : p^{\bullet}}$$

Parsing: the static/declarative method

To compute the lambda term for a net:

- assign fresh object-level variables to the hypotheses (input formulas that are not the premise of a $\times$ link or the conclusion of a $+$ link);
- assign fresh meta variables to the output literals;
- build the (partially instantiated) terms for the logical links;
- compose the matchings found at the axiom links.

This method is 'a-temporal': all the axiom links are considered simultaneously.

Remark Axiom linkings that introduce a cycle correspond to attempts to unify a metavariable with a term containing that variable (circular unification).

Parsing: the dynamic/procedural method

An alternative method to compute the lambda term for a derivation follows a path through the net. The steps of the traversal mirror the structure of the lambda term that is built up incrementally.

Travel instructions

1. enter at the (unique) terminal output formula;
2. travel upwards along output nodes until you reach an atomic formula; each $+$ you pass corresponds to a λ abstraction;
3. cross the axiom link from output to input vertex;
4. travel downwards along input nodes until you reach a hypothesis; each $\times$ you pass corresponds to an application;
5. repeat 1–4 to compute the terms for the arguments of this application.

Remark Compare this traversal with goal directed sequent proof search.

4. Abstract categorial grammar

We follow the exposition of the ESSLLI'09 course on the ACG pages.

Key idea Derive both surface forms and semantic interpretation from a more abstract source: Curry's tectogrammatical structure.

Interpretations Given source $\Sigma_1 = \langle \mathcal{A}_1, C_1, \tau_1 \rangle$, target $\Sigma_2 = \langle \mathcal{A}_2, C_2, \tau_2 \rangle$, a compositional interpretation $\mathcal{L}$ is a pair of functions $\langle \eta, \theta \rangle$ such that

- ▷ $\eta : \mathcal{A}_1 \to \mathcal{T}_{\mathcal{A}_2}$ (source atoms to target types)
- ▷ $\theta : C_1 \to \Lambda_{\Sigma_2}$ (source constants to target terms)
- ▷ $\vdash \theta(c) : \widehat{\eta}(\tau_1(c))$ (θ respects typing)

($\widehat{\eta}$ is the homomorphic extension of η: $\widehat{\eta}(\alpha \to \beta) = \widehat{\eta}(\alpha) \to \widehat{\eta}(\beta)$)

Abstract categorial grammar $\mathcal{G} = \langle \Sigma_1, \Sigma_2, \mathcal{L}, s \rangle$ (start symbol s)

- abstract language: $\textsc{source}(\mathcal{G}) = \{t \in \Lambda_{\Sigma_1} \mid \;\vdash t : s \quad \text{is derivable}\}$
- object language: $\textsc{target}(\mathcal{G}) = \{t \in \Lambda_{\Sigma_2} \mid \exists u \in \textsc{source}(\mathcal{G}). t = \mathcal{L}(u)\}$

Example: 'John seeks a unicorn'

Source signature The abstract vocabulary Σ_0: atomic types, constants, and a type-assignment function.

$$
\begin{aligned}
\Sigma_0 \quad = \quad & (\{n, np, s\}, \quad \text{(atomic types)} \\
& \{J, U, A, S\}, \quad \text{(abstract constants)} \\
& \{J \mapsto np, \quad \text{(type assignment function } \tau_0\text{)} \\
& \ U \mapsto n, \\
& \ A \mapsto n \to ((np \to s) \to s), \\
& \ S \mapsto ((np \to s) \to s) \to (np \to s) \quad \} \quad)
\end{aligned}
$$

Example (cont'd)

Target signature Concrete vocabulary Σ_1 for the surface forms (strings).
We write $string$ for the function type $* \to *$, for some arbitrary type atom $*$.

$$
\begin{aligned}
\Sigma_1 \quad = \quad & (\{*\}, \quad \text{(atomic type)} \\
& \{\mathtt{john}, \mathtt{unicorn}, \mathtt{a}, \mathtt{seeks}\}, \quad \text{(object constants)} \\
& \{\mathtt{john} \mapsto string, \quad \text{(type assignment function } \tau_1\text{)} \\
& \mathtt{unicorn} \mapsto string, \\
& \mathtt{a} \mapsto string, \\
& \mathtt{seeks} \mapsto string \quad \} \quad)
\end{aligned}
$$

Interpretation: tecto $\rightsquigarrow$ form types: $\eta(n) = \eta(np) = \eta(s) =$ *string*; constants (string concatenation is function composition: $x \cdot y =_{df} \lambda i.(x\ (y\ i))$):

$$
\begin{aligned}
\theta : \quad & J \mapsto \mathtt{john} \\
& S \mapsto \lambda p \lambda x.(p\ \lambda y.(x \cdot \mathtt{seeks} \cdot y)) \\
& A \mapsto \lambda x \lambda p.(p\ (\mathtt{a} \cdot x)) \\
& U \mapsto \mathtt{unicorn}
\end{aligned}
$$

Another interpretation

Target: meaning Σ_2 is a concrete vocabulary for modeltheoretic interpretation.

$$
\begin{aligned}
\Sigma_2 \quad = \quad & (\{e, t\}, \quad \text{(atomic types)} \\
& \{\text{J}, \text{UNICORN}, \text{SEEK}, \wedge, \exists\}, \quad \text{(object constants)} \\
& \{\text{J} \mapsto e, \quad \text{(type assignment function } \tau_2\text{)} \\
& \text{UNICORN} \mapsto e \rightarrow t, \\
& \text{SEEK} \mapsto ((e \rightarrow t) \rightarrow t) \rightarrow (e \rightarrow t), \\
& \exists \mapsto (e \rightarrow t) \rightarrow t, \\
& \wedge \mapsto t \rightarrow t \rightarrow t \quad \} \quad)
\end{aligned}
$$

Interpretation: tecto $\rightsquigarrow$ meaning types: $\eta(np) = e, \eta(n) = e \rightarrow t, \eta(s) = t$; cheating a bit for the constants ($\theta(A)$ is not linear):

$$
\begin{aligned}
\theta : \quad & J && \mapsto && \text{J} \\
& S && \mapsto && \text{SEEK} \\
& U && \mapsto && \text{UNICORN} \\
& A && \mapsto && \lambda p \lambda q.(\exists\ \lambda x.((p\ x) \wedge (q\ x)))
\end{aligned}
$$

‘John seeks a unicorn’: derivations

$$\begin{array}{ccc} & \Sigma_0 & \\ {\scriptstyle \mathcal{L}_{0,1}} \swarrow & & \searrow {\scriptstyle \mathcal{L}_{0,2}} \\ \Sigma_1 & & \Sigma_2 \end{array}$$

Abstract terms $t_1 : (S\ (A\ U)\ J)$; $\quad t_2 : (A\ U\ \lambda x.(S\ \lambda k.(k\ x))\ J))$

Interpretation: form $\mathcal{L}_{0,1}(t_1) = \mathcal{L}_{0,1}(t_2) = \texttt{john} \cdot \texttt{seeks} \cdot \texttt{a} \cdot \texttt{unicorn}$

Interpretation: meaning

$$\begin{aligned} \mathcal{L}_{0,2}(t_1) &= (\textsc{seek}\ \lambda q.(\exists\ \lambda x.(\textsc{unicorn}\ x) \wedge (q\ x))\ \textsc{j}) \\ \mathcal{L}_{0,2}(t_2) &= (\exists\ \lambda x.(\textsc{unicorn}\ x) \wedge (\textsc{seek}\ \lambda p.(p\ x)\ \textsc{j})) \end{aligned}$$

- each of the interpretations (form, meaning) are compositional homomorphisms
- the relation form $\rightsquigarrow$ meaning is not: one surface form, two meanings

ACG complexity hierarchy

A hierarchy of ACG's is obtained in terms of two parameters:

- complexity of the abstract structures: maximal order of its constants
- complexity of the interpretation: maximal order of the image of source atoms

Complexity of abstract signature

$$\mathsf{ord}(\alpha) = 1, \ \alpha \text{ atomic}; \qquad \mathsf{ord}(\alpha \to \beta) = \max(\mathsf{ord}(\alpha) + 1, \mathsf{ord}(\beta))$$

$$\mathsf{ord}(\Sigma) = \max_{c \in C}(\mathsf{ord}(\tau(c)))$$

Complexity of interpretation

$\Sigma_1 = (A_1, C_1, \tau)$, $\Sigma_2 = (A_2, C_2, \tau)$, $\mathcal{L} : \Sigma_1 \to \Sigma_2$

$$\mathsf{compl}(\mathcal{L}) = \max_{\alpha \in A_1}(\mathsf{ord}(\mathcal{L}(\alpha)))$$

Abstract categorial hierarchy

Grammars For $\mathcal{G} = (\Sigma_1, \Sigma_2, \mathcal{L}, s)$, let order$(\mathcal{G})$ = ord(Σ_1), complexity$(\mathcal{G})$ = compl$(\mathcal{L})$

$$\mathsf{G}(m,n) = \{\mathcal{G} \mid \text{order}(\mathcal{G}) \leq m,\ \text{complexity}(\mathcal{G}) \leq n\}$$

Languages

$$\mathsf{L}(m,n) = \{\textsc{target}(\mathcal{G}) \mid \mathcal{G} \in \mathsf{G}(m,n)\}$$

String languages for $\mathsf{L}(2,n)$

ACG type	language
$\mathsf{L}(2,1)$	regular
$\mathsf{L}(2,2)$	context-free
$\mathsf{L}(2,3)$	well-nested mildly context sensitive
$\mathsf{L}(2,4)$	mildly context sensitive
$\mathsf{L}(2,4+n)$	$= \mathsf{L}(2,4)$

Illustration: context-free grammars

Recall: a context-free grammar G is a 4-tuple (V, Σ, R, S), where

V is an alphabet,

Σ (the set of terminals) is a subset of V,

R (the set of rules) is a finite subset of $(V - \Sigma) \times V^*$, and

S (the start symbol) is an element of $V - \Sigma$.

The members of $V - \Sigma$ are called nonterminals.

ACG encoding of context-free grammars

Example Well-nested bracketing for a bracket pair a, $\overline{a}$.

$$
\begin{array}{ll}
S \longrightarrow a\ S\ \overline{a} & R_1 : S \to S \\
S \longrightarrow S\ S & R_2 : S \to S \to S \\
S \longrightarrow \epsilon & R_3 : S
\end{array}
$$

Source non-terminal symbols $\rightsquigarrow$ types; rules $\rightsquigarrow$ abstract constants.

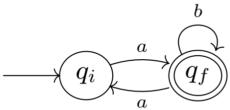
Target type: $string$; constants: terminal symbols.

Interpretation types: $\eta(S) = string$; constants:

$$
\theta : \quad
\begin{array}{lcl}
R_1 & : & \lambda x.(a \cdot x \cdot \overline{a}) \\
R_2 & : & \lambda x \lambda y.(x \cdot y) \\
R_3 & : & \lambda x.x
\end{array}
$$

Illustration: Finite state automata

Example



Source states $\rightsquigarrow$ atomic types; transitions $\rightsquigarrow$ constants, with special constant for final state. $\Sigma_1 = (\{q_i, q_f\}, \{t_0, t_1, t_2, t_3\}, \tau_1)$, where τ_1 is 'reading backwards':

$$t_0 \mapsto q_f \qquad t_1 \mapsto q_f \to q_i \qquad t_2 \mapsto q_f \to q_f \qquad t_3 \mapsto q_i \to q_f$$

Target Σ_2 has one atomic type: σ; end-of-string $\# : \sigma$; alphabet symbols: $\sigma \to \sigma$

Interpretation Start symbol: q_i. $\mathcal{L} : \Sigma_1 \to \Sigma_2$, with $\eta(q_i) = \eta(q_f) = \sigma$. Constants:

$$\theta : \quad \begin{array}{lcl} t_0 & \mapsto & \# \\ t_1 & \mapsto & \lambda x.(\mathsf{a}\ x) \\ t_2 & \mapsto & \lambda x.(\mathsf{b}\ x) \\ t_3 & \mapsto & \lambda x.(\mathsf{a}\ x) \end{array}$$

Next class: encoding formalisms beyond context-free (TAG, k-MCFG).

$\square$