Computation as Subtyping

On the Turing Completeness of Type Systems, with Applications to Formal Grammars

Guy Emerson

Update

- Draft paper!
  `https://www.cl.cam.ac.uk/~gete2/wrapper.pdf`

Update

- Draft paper!
  `https://www.cl.cam.ac.uk/~gete2/wrapper.pdf`
- Simplified constructions/proofs
- Best practices, with examples
- Detailed discussion (comparison with "junk slots", two kinds of input, "currying" for multiple arguments, nondeterministic computation)

Recursion: Pathological Counterexample

$$\begin{tikzcd} & a \arrow[dl, dash] \arrow[dr, dash] & \\ b \arrow[dr, dash] & & x \arrow[dl, dash] \\ & y & \end{tikzcd}$$

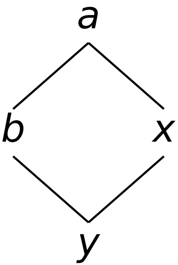
a

$b \xrightarrow{F} a$

x

$y \xrightarrow{F} b \xrightarrow{F} x$

Recursion: Pathological Counterexample



$$a$$

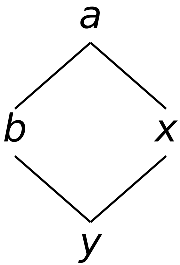
$$b \xrightarrow{F} a$$

$$x$$

$$y \xrightarrow{F} b \xrightarrow{F} x$$

$$y \xrightarrow{F} b \xrightarrow{F} x \quad \sqcap \quad b \xrightarrow{F} x$$

Recursion: Pathological Counterexample



$$a$$

$$b \xrightarrow{\mathsf{F}} a$$

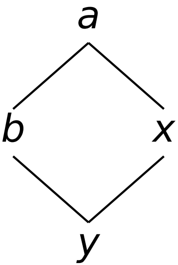
$$x$$

$$y \xrightarrow{\mathsf{F}} b \xrightarrow{\mathsf{F}} x$$

$$y \xrightarrow{\mathsf{F}} b \xrightarrow{\mathsf{F}} x \quad \sqcap \quad b \xrightarrow{\mathsf{F}} x$$

$$= \quad y \xrightarrow{\mathsf{F}} y \xrightarrow{\mathsf{F}} x$$

Recursion: Pathological Counterexample



$$a$$

$$b \xrightarrow{F} a$$

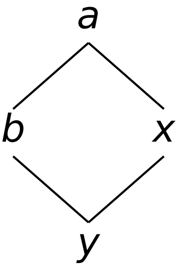
$$x$$

$$y \xrightarrow{F} b \xrightarrow{F} x$$

$$y \xrightarrow{F} b \xrightarrow{F} x \quad \sqcap \quad b \xrightarrow{F} x$$

$$= \quad y \xrightarrow{F} y \xrightarrow{F} y \xrightarrow{F} x$$

Recursion: Pathological Counterexample



$$a$$

$$b \xrightarrow{\mathsf{F}} a$$

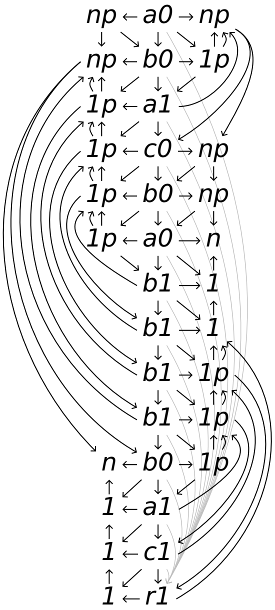
$$x$$

$$y \xrightarrow{\mathsf{F}} b \xrightarrow{\mathsf{F}} x$$

$$\begin{aligned} & y \xrightarrow{\mathsf{F}} b \xrightarrow{\mathsf{F}} x \quad \sqcap \quad b \xrightarrow{\mathsf{F}} x \\ = \quad & y \xrightarrow{\mathsf{F}} y \xrightarrow{\mathsf{F}} y \xrightarrow{\mathsf{F}} y \xrightarrow{\mathsf{F}} \ldots \end{aligned}$$

$$
\begin{tikzcd}
str & a \arrow[l] \arrow[r] \arrow[dd] \arrow[dr] \arrow[ddd, bend left] & str & \\
 & & sym \arrow[r] & str \\
str & mach \arrow[l] \arrow[r] & str & \\
str & mach \arrow[l] \arrow[r] & str &
\end{tikzcd}
\quad \sqcap \quad n \leftarrow r0 \rightarrow n \quad =
$$

$$\begin{tikzcd}[column sep=small, row sep=tiny]
 & str & a \arrow[l] \arrow[r] \arrow[dr] \arrow[dd] \arrow[ddd, bend left, gray] & str & \\
 & & & sym \arrow[r] & str \\
 & str & mach \arrow[l] \arrow[r] & str & \\
 & str & mach \arrow[l] \arrow[r] & str &
\end{tikzcd}
\quad \sqcap \quad n \longleftarrow r0 \longrightarrow n \quad = $$



Theorem 1

- For any FSA, there is a one-feature type system where unification can determine whether the FSA accepts a string
- For any one-feature type system, there is an FSA which recognises when two feature structures are unifiable

Theorem 2

- For any Turing machine, there is a two-feature type system where unification can determine whether the Turing machine halts on a given input

“Programming interface”

$$\begin{bmatrix} \textit{my-phrase-type} & \\ \text{MY-PATH} & \begin{bmatrix}\text{AND} & \langle \boxed{1}, \boxed{2} \rangle\end{bmatrix} \\ \text{HEAD-DTR}|\text{MY-PATH} & \begin{bmatrix}\text{BOOL} & \boxed{1}\end{bmatrix} \\ \text{NON-HEAD-DTR}|\text{MY-PATH} & \begin{bmatrix}\text{BOOL} & \boxed{2}\end{bmatrix}\end{bmatrix}$$

"Programming interface"

$$\begin{bmatrix} \textit{my-phrase-type} & \\ \text{MY-PATH} & \begin{bmatrix}\text{AND} \ \langle \boxed{1}, \boxed{2} \rangle\end{bmatrix} \\ \text{HEAD-DTR}|\text{MY-PATH} & \boxed{1} \\ \text{NON-HEAD-DTR}|\text{MY-PATH} & \boxed{2} \end{bmatrix}$$

Wrapper types as input?

"Programming interface"

$$\begin{bmatrix} \textit{my-phrase-type} & \\ \text{MY-PATH} & \begin{bmatrix}\text{AND} \ \left\langle \begin{bmatrix}\text{BOOL} \ \boxed{1}\end{bmatrix}, \begin{bmatrix}\text{BOOL} \ \boxed{2}\end{bmatrix} \right\rangle\end{bmatrix} \\ \text{HEAD-DTR}|\text{MY-PATH} & \begin{bmatrix}\text{BOOL} \ \boxed{1}\end{bmatrix} \\ \text{NON-HEAD-DTR}|\text{MY-PATH} & \begin{bmatrix}\text{BOOL} \ \boxed{2}\end{bmatrix} \end{bmatrix}$$

Wrapper types as input, cutting off computation history

“Programming interface”

$$\begin{bmatrix} \textit{my-phrase-type} & \\ \text{MY-PATH} & \begin{bmatrix}\text{AND} & \langle \boxed{1}, \boxed{2} \rangle\end{bmatrix} \\ \text{HEAD-DTR}|\text{MY-PATH} & \begin{bmatrix}\text{BOOL} & \boxed{1}\end{bmatrix} \\ \text{NON-HEAD-DTR}|\text{MY-PATH} & \begin{bmatrix}\text{BOOL} & \boxed{2}\end{bmatrix} \end{bmatrix}$$

Data types as input: never have computation history, but also never allow composition of wrappers in one rule

Practical Examples

- Logical operations (negation, and, or)
  - Application: coordination
- List operations (append, nondeterministic pop)
  - Application: long-distance dependencies
  - Application: valence changes
  - Application: flexible word order

Deterministic head-comp rules

$$\begin{bmatrix} \textit{head-1st-comp-phrase} & \\ \text{SYNSEM}|\text{L}|\text{CAT}|\text{VAL}|\text{COMPS} & \boxed{2} \\ \text{HEAD-DTR}|\text{SYNSEM}|\text{L}|\text{CAT}|\text{VAL}|\text{COMPS} & \begin{bmatrix} \text{FIRST} & \boxed{1} \\ \text{REST} & \boxed{2} \end{bmatrix} \\ \text{NON-HEAD-DTR}|\text{SYNSEM} & \boxed{1} \end{bmatrix}$$

Deterministic head-comp rules

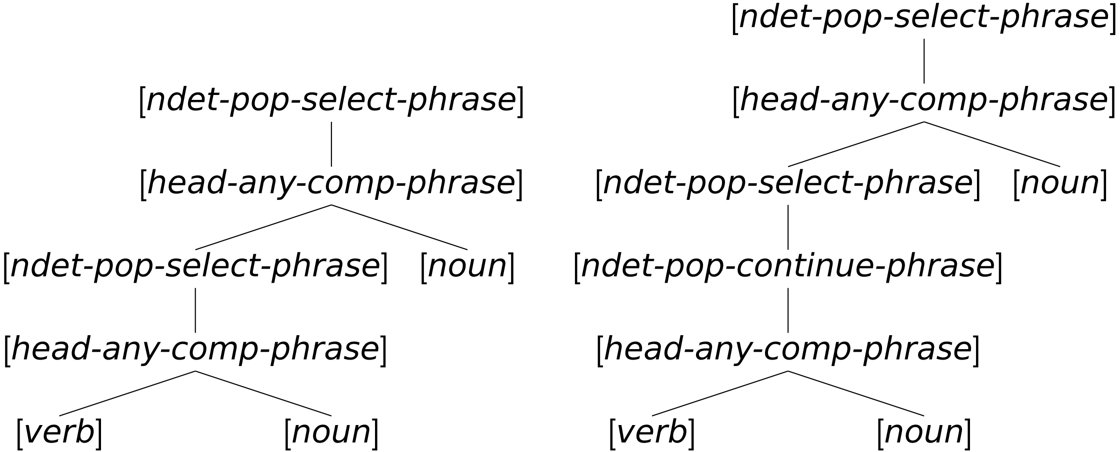
$$
\begin{bmatrix}
\textit{head-1st-comp-phrase} & \\
\text{SYNSEM}|\text{L}|\text{CAT}|\text{VAL}|\text{COMPS} & \boxed{2} \\
\text{HEAD-DTR}|\text{SYNSEM}|\text{L}|\text{CAT}|\text{VAL}|\text{COMPS} & \begin{bmatrix} \text{FIRST} & \boxed{1} \\ \text{REST} & \boxed{2} \end{bmatrix} \\
\text{NON-HEAD-DTR}|\text{SYNSEM} & \boxed{1}
\end{bmatrix}
$$

$$
\begin{bmatrix}
\textit{head-2nd-comp-phrase} & \\
\text{SYNSEM}|\text{L}|\text{CAT}|\text{VAL}|\text{COMPS} & \begin{bmatrix} \text{FIRST} & \boxed{1} \\ \text{REST} & \boxed{3} \end{bmatrix} \\
\text{HEAD-DTR}|\text{SYNSEM}|\text{L}|\text{CAT}|\text{VAL}|\text{COMPS} & \begin{bmatrix} \text{FIRST} & \boxed{1} \\ \text{REST} & \begin{bmatrix} \text{FIRST} & \boxed{2} \\ \text{REST} & \boxed{3} \end{bmatrix} \end{bmatrix} \\
\text{NON-HEAD-DTR}|\text{SYNSEM} & \boxed{2}
\end{bmatrix}
$$

Nondeterministic head-comp rule

$$\begin{bmatrix} \textit{head-any-comp-phrase} & \\ \text{SYNSEM}|\text{L}|\text{CAT}|\text{VAL}|\text{COMPS} & \boxed{2} \\ \text{HEAD-DTR}|\text{SYNSEM}|\text{L}|\text{CAT}|\text{VAL}|\text{COMPS} & \boxed{1} \\ \text{NON-HEAD-DTR}|\text{SYNSEM} & \boxed{3} \\ \text{NDET} \quad \begin{bmatrix} \text{POP-INPUT} & \boxed{1} \\ \text{POP-OUTPUT-LIST} & \boxed{2} \\ \text{POP-OUTPUT-ITEM} & \boxed{3} \end{bmatrix} & \end{bmatrix}$$

Word order ambiguity



Summary

- Relational constraints are possible and practical
- Both deterministic and nondeterministic
- Feedback welcome!
  `https://www.cl.cam.ac.uk/~gete2/wrapper.pdf`