

Computation as Subtyping

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

Guy Emerson

Update

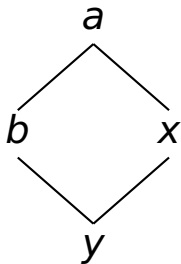
- Draft paper!

<https://www.cl.cam.ac.uk/~gete2/wrapper.pdf>

Update

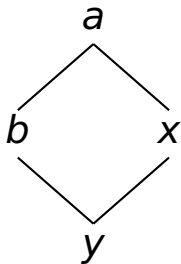
- Draft paper!
<https://www.cl.cam.ac.uk/~gete2/wrapper.pdf>
- Simplified constructions/proofs
- Best practices, with examples
- Detailed discussion (comparison with “junk slots”, two kinds of input, “currying” for multiple arguments, nondeterministic computation)

Recursion: Pathological Counterexample



$$\begin{array}{l} a \\ b \xrightarrow{F} a \\ x \\ y \xrightarrow{F} b \xrightarrow{F} x \end{array}$$

Recursion: Pathological Counterexample

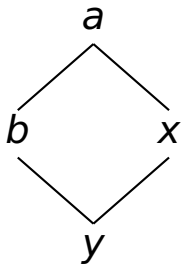


$$a$$
$$b \xrightarrow{F} a$$

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

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

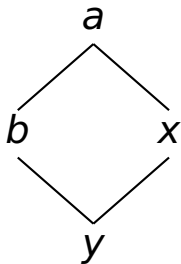
Recursion: Pathological Counterexample



$$\begin{array}{l} a \\ b \xrightarrow{F} a \\ x \\ y \xrightarrow{F} b \xrightarrow{F} x \end{array}$$

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

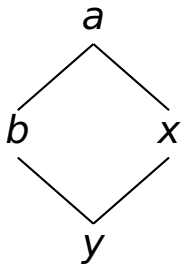
Recursion: Pathological Counterexample



$$\begin{array}{l} a \\ b \xrightarrow{F} a \\ x \\ y \xrightarrow{F} b \xrightarrow{F} x \end{array}$$

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

Recursion: Pathological Counterexample

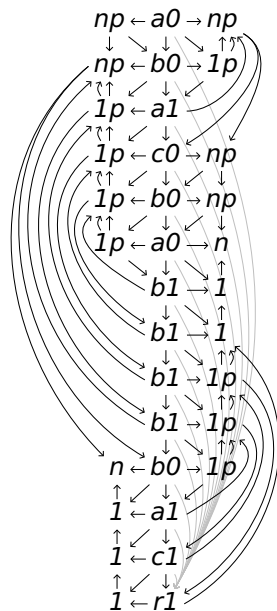


$$\begin{array}{l} a \\ b \xrightarrow{F} a \\ x \\ y \xrightarrow{F} b \xrightarrow{F} x \end{array}$$

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

$$\begin{array}{l}
 str \leftarrow a \rightarrow str \\
 \quad \downarrow \searrow \\
 str \leftarrow mach \rightarrow str \quad sym \rightarrow str \\
 str \leftarrow mach \rightarrow str
 \end{array}
 \quad \sqcap \quad n \leftarrow r0 \rightarrow n \quad =$$

$$\begin{array}{l}
 str \leftarrow a \rightarrow str \\
 \quad \downarrow \quad \searrow \text{sym} \rightarrow str \\
 str \leftarrow mach \rightarrow str \\
 str \leftarrow mach \rightarrow str
 \end{array}$$

$$\sqcap \quad n \leftarrow r0 \rightarrow n \quad =$$


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”

<i>my-phrase-type</i>	
MY-PATH	[AND <[1], [2]>]
HEAD-DTR MY-PATH	[BOOL [1]]
NON-HEAD-DTR MY-PATH	[BOOL [2]]

“Programming interface”

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

Wrapper types as input?

“Programming interface”

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

Data types as input: less computation history

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

<i>head-1st-comp-phrase</i>		
SYNSEM L CAT VAL COMPS	<div>2</div>	
HEAD-DTR SYNSEM L CAT VAL COMPS	<div>FIRST</div>	<div>1</div>
	<div>REST</div>	<div>2</div>
NON-HEAD-DTR SYNSEM	<div>1</div>	

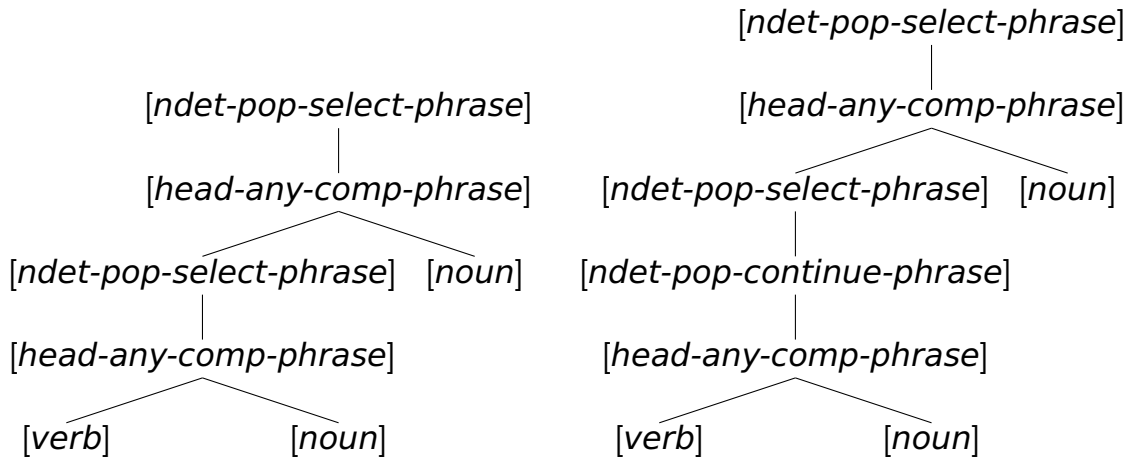
Deterministic head-comp rules

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

Nondeterministic head-comp rule

<i>head-any-comp-phrase</i>			
SYNSEM L CAT VAL COMPS			2
HEAD-DTR SYNSEM L CAT VAL COMPS			1
NON-HEAD-DTR SYNSEM			3
NDET	POP-INPUT	1	
	POP-OUTPUT-LIST	2	
	POP-OUTPUT-ITEM	3	

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>