Weighted Attribute Grammars:

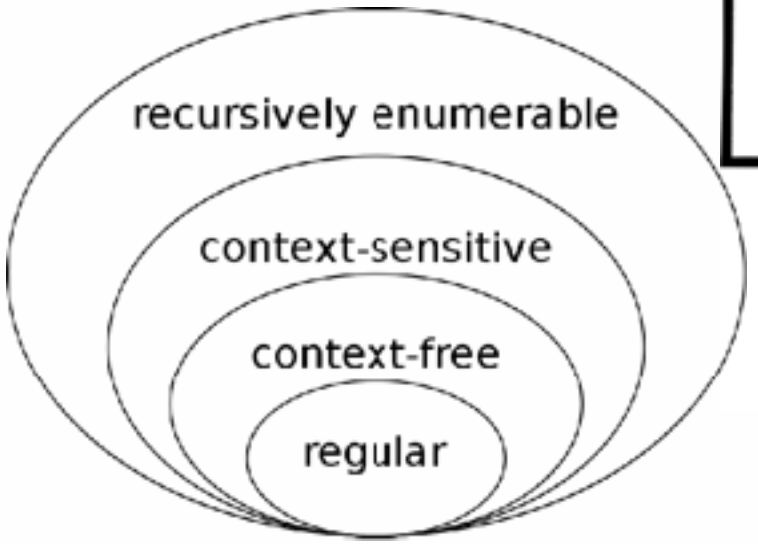
Reconciling Weighted and Attribute Grammars

WADT @ STAF 2026, 30 June

Marcus Gerhold, Vadim Zaytsev, University of Twente, 🇳🇱

UNIVERSITY OF TWENTE. | Formal Methods & Tools

Context-Free Grammars



Weighted/Probabilistic Grammars

- Each production rule is annotated with
  - weight ⇒ looking for minimum total
    - not just CFG
  - probability ⇒ seeking most likely parse
    - can steer generation
- Many applications
  - from quantitative constraint logic programming
  - to procedural content generation

Attribute/Affix Grammars

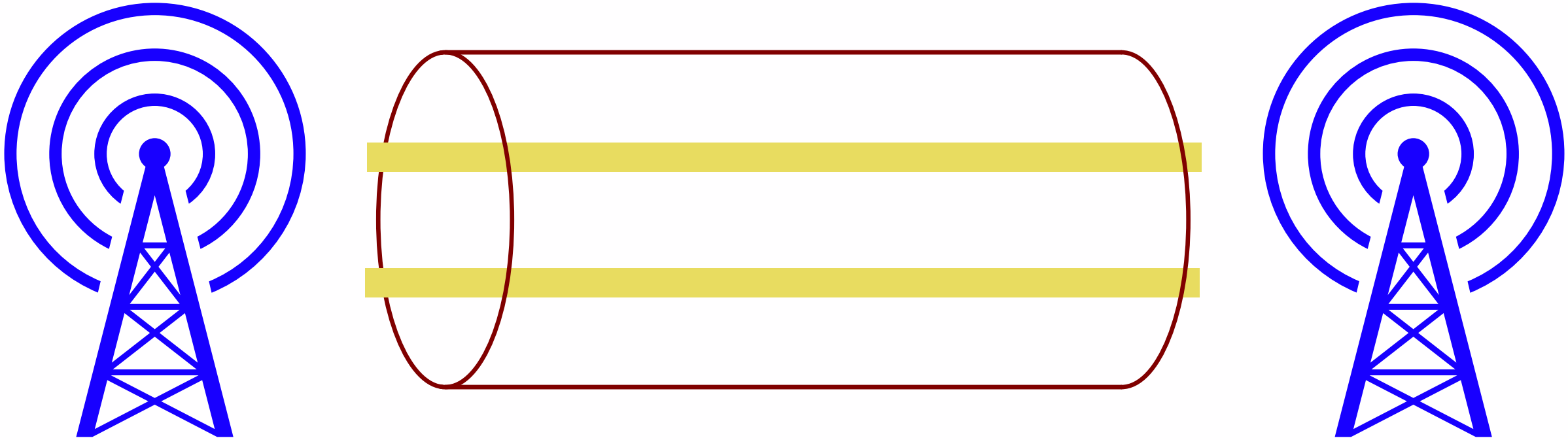
- Augment CFG with attributes/variables
  - inherited: propagate downwards to children
  - synthesised: propagate upwards to the parent
  - . . .
- Actively researched since the 1960s
- Can graciously handle many context-sensitive tasks
- cf. Data-Dependent Grammars [POPL2010]

Weighted Attribute Grammars

- Nonterminals may have attributes
- Production rules may have weights
- Weights can be expressions
- Expressions can include attributes
- Opens many doors
- Trivial combinations are easy
- Deeper interplay demands further research

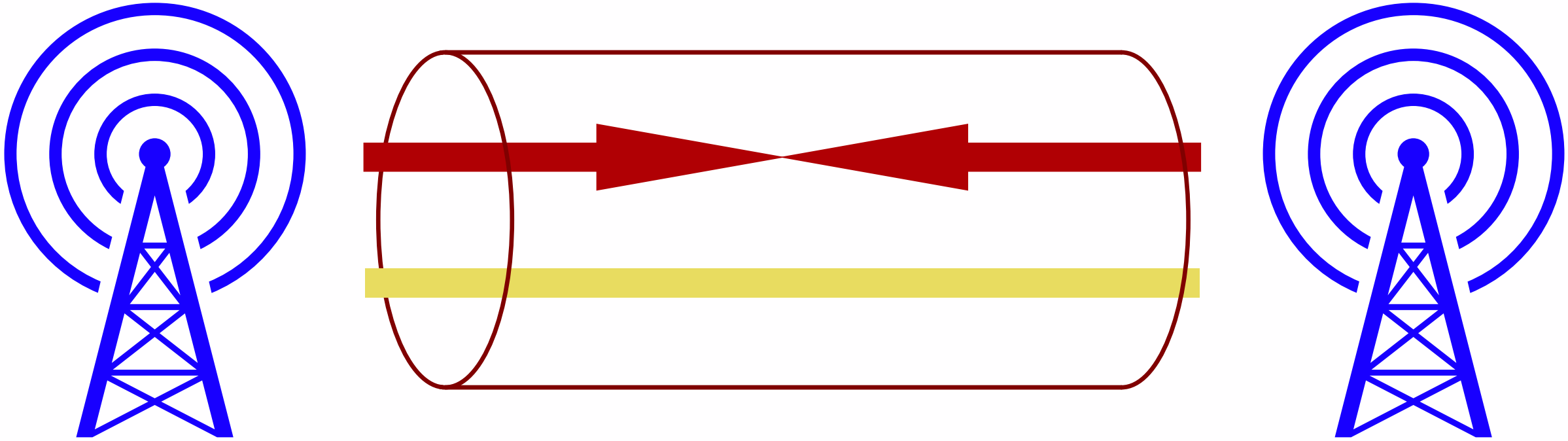
BEB: Binary Exponential Backoff

BEB: Binary Exponential Backoff

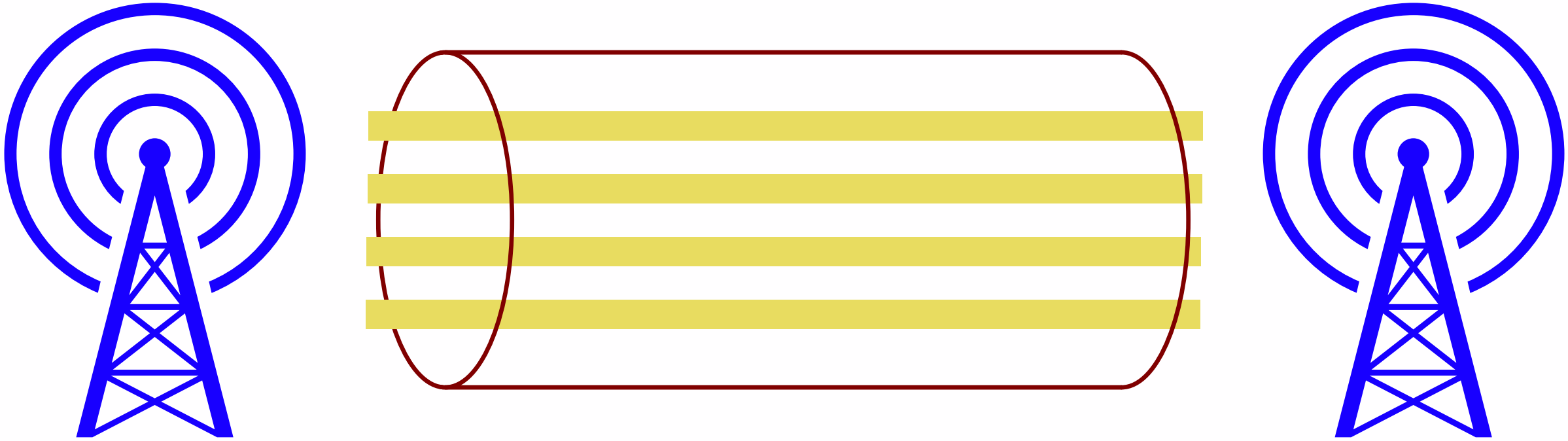


BEB: Binary Exponential Backoff

BEB: Binary Exponential Backoff

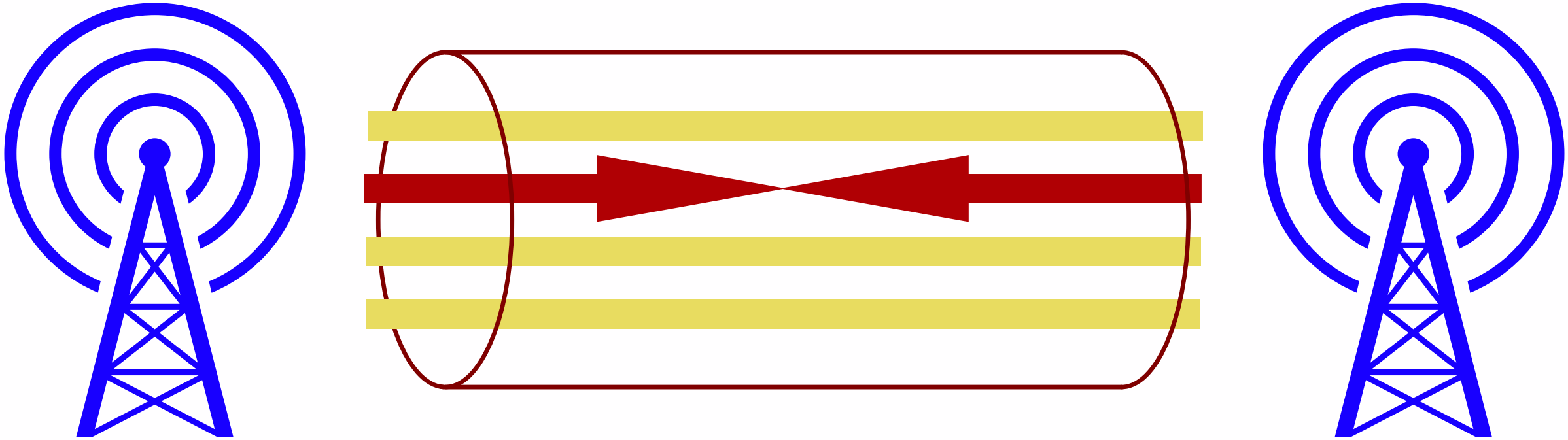


BEB: Binary Exponential Backoff

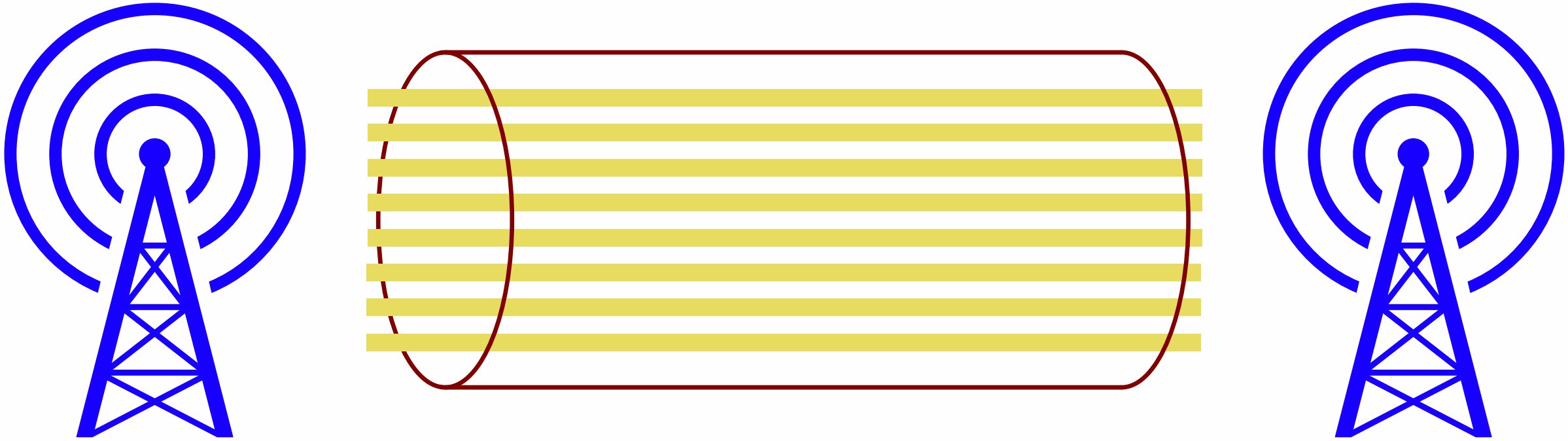


BEB: Binary Exponential Backoff

BEB: Binary Exponential Backoff



BEB: Binary Exponential Backoff



BEB in Text

```
  [1/4]  1,2.
  [1/4]  2,1.
 [1/64]  1,1;1,2.
         . . .
 [1/64]  2,2;4,3.
         . . .
[1/4096] 1,1;1,1;1,2.
         . . .
```

BEB in CFG (Truncated)

$$
\begin{aligned}
S &\longrightarrow P_1 \\
P_1 &\longrightarrow \texttt{'1,1;'}\, P_2 \\
P_1 &\longrightarrow \texttt{'1,2.'} \\
P_1 &\longrightarrow \texttt{'2,1.'} \\
P_1 &\longrightarrow \texttt{'2,2;'}\, P_2
\end{aligned}
$$

$$
1 + \sum_{i=1}^{n} 2^{2i} = \sum_{i=0}^{n} 4^i = \frac{4^{n+1} - 1}{3}
$$

$$
\begin{aligned}
P_2 &\longrightarrow \texttt{'1,1;'}\, P_4 \\
P_2 &\longrightarrow \texttt{'1,2.'} \\
P_2 &\longrightarrow \texttt{'1,3.'} \\
P_2 &\longrightarrow \texttt{'1,4.'} \\
P_2 &\longrightarrow \texttt{'2,1.'} \\
P_2 &\longrightarrow \texttt{'2,2;'}\, P_4 \\
P_2 &\longrightarrow \texttt{'2,3.'} \\
P_2 &\longrightarrow \texttt{'2,4.'} \\
P_2 &\longrightarrow \texttt{'3,1.'} \\
P_2 &\longrightarrow \texttt{'3,2.'} \\
P_2 &\longrightarrow \texttt{'3,3;'}\, P_4 \\
P_2 &\longrightarrow \texttt{'3,4.'} \\
P_2 &\longrightarrow \texttt{'4,1.'} \\
P_2 &\longrightarrow \texttt{'4,2.'} \\
P_2 &\longrightarrow \texttt{'4,3.'} \\
P_2 &\longrightarrow \texttt{'4,4;'}\, P_4
\end{aligned}
$$

BEB in PCFG (Truncated)

$$S \xrightarrow{1} P_1$$

$$P_1 \xrightarrow{1/4} \texttt{'1,1;'}\, P_2$$

$$P_1 \xrightarrow{1/4} \texttt{'1,2.'}$$

$$P_1 \xrightarrow{1/4} \texttt{'2,1.'}$$

$$P_1 \xrightarrow{1/4} \texttt{'2,2;'}\, P_2$$

$$1 + \sum_{i=1}^{n} 2^{2i} = \sum_{i=0}^{n} 4^i = \frac{4^{n+1} - 1}{3}$$

$$P_2 \xrightarrow{1/16} \texttt{'1,1;'}\, P_4$$
$$P_2 \xrightarrow{1/16} \texttt{'1,2.'}$$
$$P_2 \xrightarrow{1/16} \texttt{'1,3.'}$$
$$P_2 \xrightarrow{1/16} \texttt{'1,4.'}$$
$$P_2 \xrightarrow{1/16} \texttt{'2,1.'}$$
$$P_2 \xrightarrow{1/16} \texttt{'2,2;'}\, P_4$$
$$P_2 \xrightarrow{1/16} \texttt{'2,3.'}$$
$$P_2 \xrightarrow{1/16} \texttt{'2,4.'}$$
$$P_2 \xrightarrow{1/16} \texttt{'3,1.'}$$
$$P_2 \xrightarrow{1/16} \texttt{'3,2.'}$$
$$P_2 \xrightarrow{1/16} \texttt{'3,3;'}\, P_4$$
$$P_2 \xrightarrow{1/16} \texttt{'3,4.'}$$
$$P_2 \xrightarrow{1/16} \texttt{'4,1.'}$$
$$P_2 \xrightarrow{1/16} \texttt{'4,2.'}$$
$$P_2 \xrightarrow{1/16} \texttt{'4,3.'}$$
$$P_2 \xrightarrow{1/16} \texttt{'4,4;'}\, P_4$$

BEB in AG

$$
\begin{aligned}
S &\longrightarrow Seq(1) \\
Seq(\downarrow n) &\longrightarrow Pair(n, l, r)\ Eq(l \stackrel{?}{=} r, n) \\
Eq(\downarrow c, \downarrow n) &\longrightarrow \langle c? \rangle\ \text{';'}\ Seq(n+1) \\
Eq(\downarrow c, \downarrow n) &\longrightarrow \langle \neg c? \rangle\ \text{'.'} \\
Pair(\downarrow n, \uparrow l, \uparrow r) &\longrightarrow Number(n, l)\ \text{','}\ Number(n, r) \\
Number(\downarrow n, \uparrow x) &\longrightarrow Bits(n, 0, x)\ x \\
Bits(\downarrow n, \downarrow a, \uparrow x) &\longrightarrow \langle n \stackrel{?}{=} 0 \rangle\ \varepsilon\ \langle x := a + 1 \rangle \\
Bits(\downarrow n, \downarrow a, \uparrow x) &\longrightarrow \langle n \stackrel{?}{\neq} 0 \rangle\ Bit(b)\ Bits(n-1, 2a+b, y)\ \langle x := y \rangle \\
Bit(\uparrow v) &\longrightarrow \varepsilon\ \langle v := 0 \rangle \\
Bit(\uparrow v) &\longrightarrow \varepsilon\ \langle v := 1 \rangle
\end{aligned}
$$

BEB in WAG

$$
\begin{aligned}
S &\xrightarrow{1} P\{[1,1]\} \\
P\{\downarrow v\} &\xrightarrow{1} E\{v,l\}\ E\{v,r\}\ C\{v,l,r\} \\
E\{\downarrow v, \uparrow r\} &\xrightarrow{v_i/|v|} \varepsilon\ \langle r := i \rangle \\
C\{\downarrow v, \downarrow x, \downarrow y\} &\xrightarrow{x=y} x\text{','}y\text{';'}\ P\{\texttt{dup}(v)\} \\
C\{\downarrow v, \downarrow x, \downarrow y\} &\xrightarrow{x\neq y} x\text{','}y\text{'.'}
\end{aligned}
$$

The "GOLF" Definition

Definition 1. *A weighted attribute grammar (WAG) is a tuple* $\mathcal{G} = \langle \Gamma, \Omega, \Lambda, \Phi \rangle$*, where*

- $\Gamma = \langle N, T, P, s \rangle$ *is a tuple of foundational components of a Chomskyan grammar;*
  - N *is a set of nonterminal symbols (a vocabulary), denoted in blue, as keywords in programming languages*[1]*;*
  - T *is a set of terminals (an alphabet),* $T \cap N = \varnothing$*, quoted and written in* `'red'` *like string constants in programming languages;*
  - P *is a set of production rules, appropriately restricted depending on the actual grammar type: regular, context-free, etc;*
  - $s \in N$ *is the starting nonterminal;*
- $\Omega = \langle \omega, \beta \rangle$ *is a tuple of weight-related components;*
  - $\omega : P \to \mathbb{T}$ *assigns types to weights of production rules; weights that appear in the same context (i.e., in different production rules of the same nonterminal), must be of comparable type.*
  - β *assigns weights to each production rule; the domain of* β *is* P*, but the codomain depends on the input such that* $\beta(p) \in \omega(p) \cup A$*;*
- $\Lambda = \langle A, \tau, \kappa, \pi, \sigma \rangle$ *is a tuple of attribute components;*
  - A *is a set of attributes;*
  - $\tau : A \to \mathbb{T}$ *assigns types to attributes;*
  - $\kappa : N \to A^\star$ *specifies inherited attributes for each nonterminal, propagated from instantiating nodes to their children; such attributes will be denoted with a downward arrow:* $\downarrow x$*;*
  - $\pi : N \to A^\star$ *specifies synthesised attributes for each nonterminal, propagated from instantiated nodes to their parents; such attributes will be denoted with an upward arrow:* $\uparrow x$*;*
  - $\sigma : N \to A^\star$ *specifies "static" attributes that are shared among all nodes of the same nonterminal; they do not exist in classic attribute grammars so borrowing UML's notation, we will underline them:* $\underline{x}$*;*
- Φ *are computation formulae, such as* $\langle x := 0 \rangle$*.* ■

Conclusion

- WAG ::= (WG | PCFG) & AG
- Many applications: MBT, DDA, PCG, CLP, SR…
  - Binary Exponential Backoff shown
- Good? elegantly express the intent
- Bad? everything needs to be proven from ground up
- Ugly? tool support does a lot of heavy lifting
- **Questions? Pointers? Advice?**