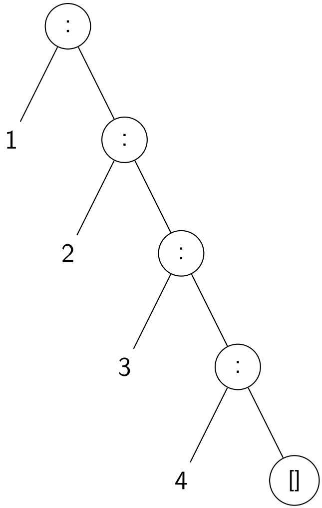
Efficient Divide-and-Conquer Parsing of Practical Context-Free Languages

Jean-Philippe Bernardy   Koen Claessen

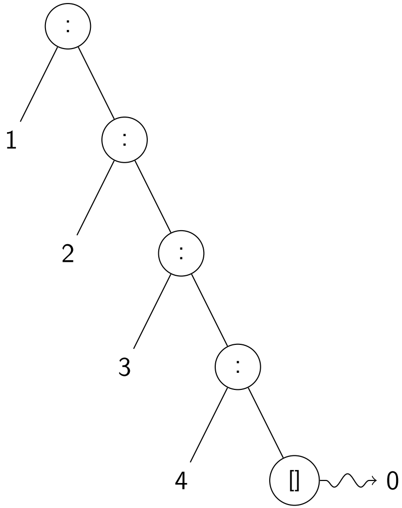
CHALMERS | GÖTEBORG UNIVERSITY

CLASP Seminar, Nov 14, 2016

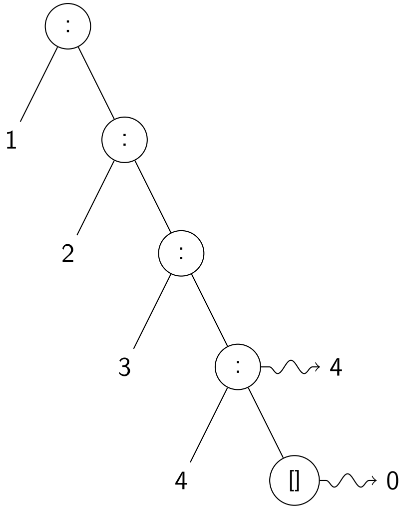
FP workhorse: lists



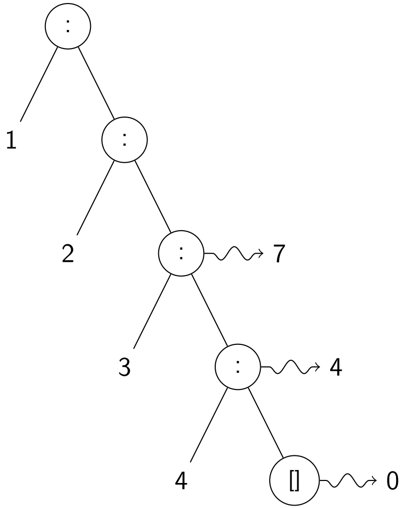
FP workhorse: lists



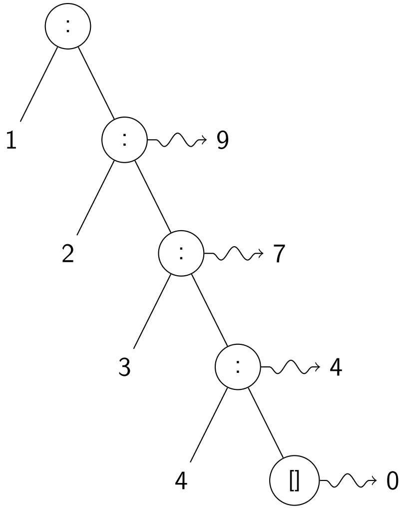
FP workhorse: lists



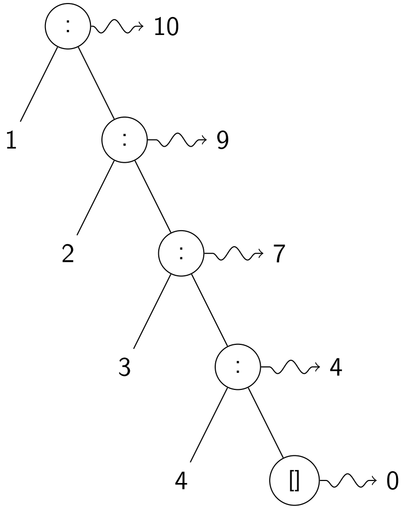
FP workhorse: lists



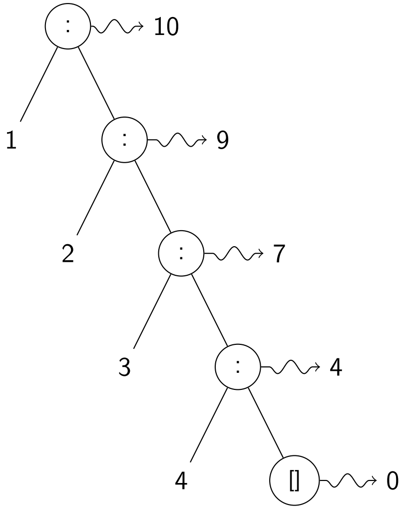
FP workhorse: lists



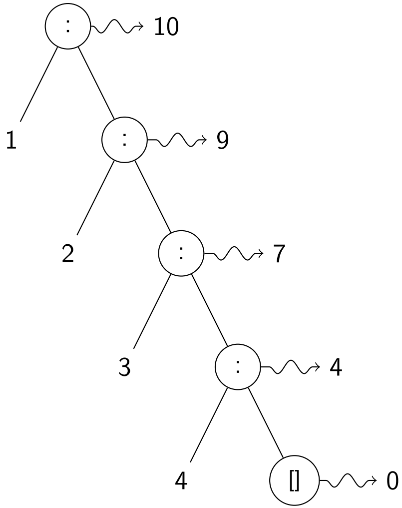
FP workhorse: lists



FP workhorse: lists

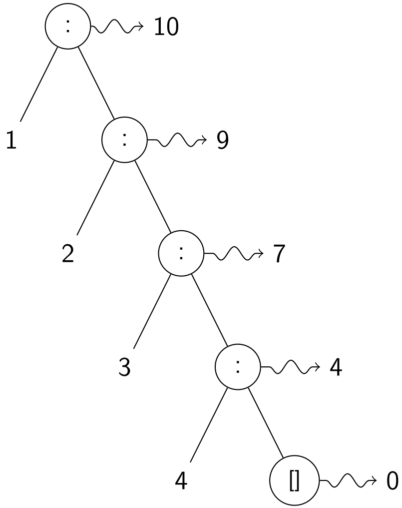


FP workhorse: lists



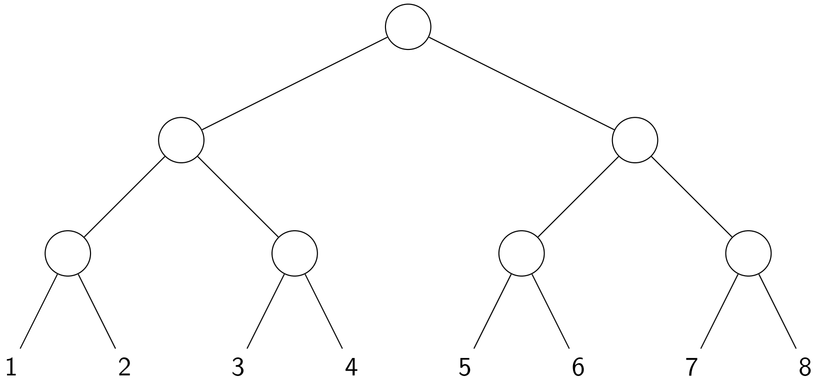
- Built-in sequentiality

FP workhorse: lists

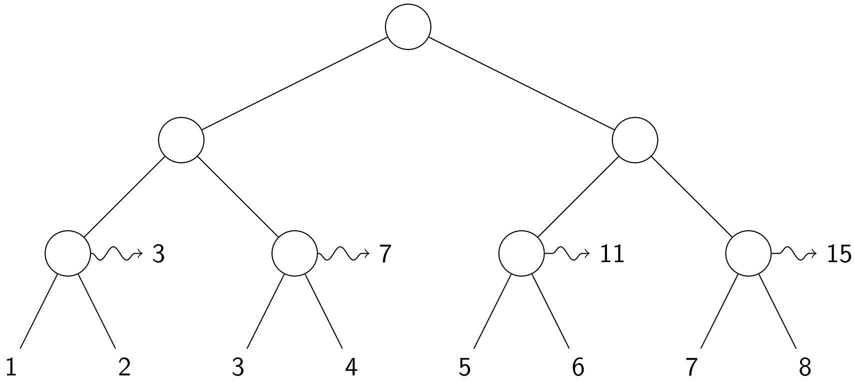


- Built-in sequentiality
- Bad!

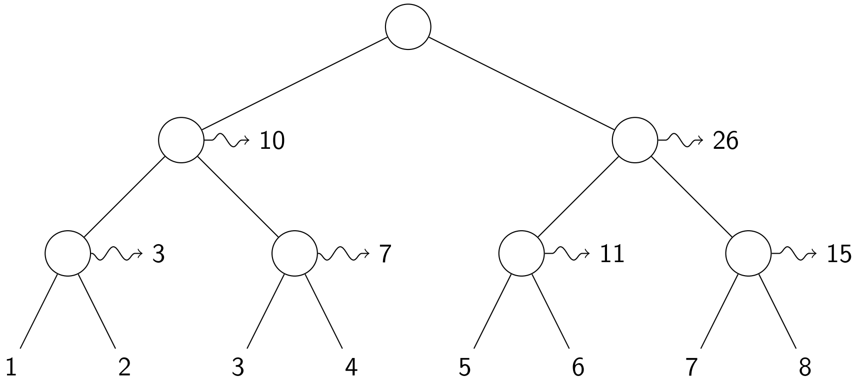
Exploiting parallelism: sum over a tree



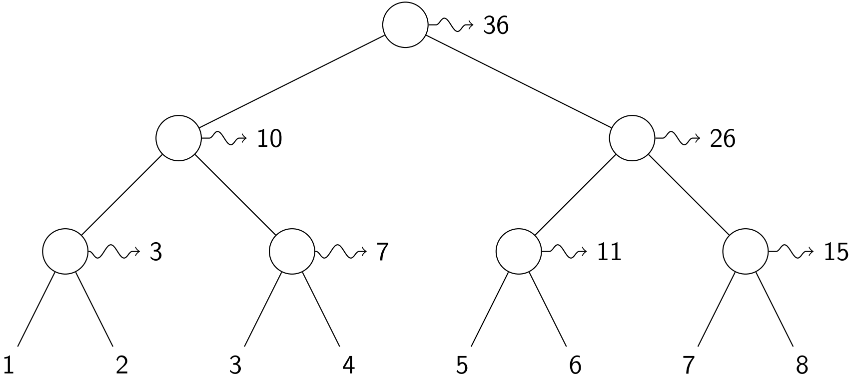
Exploiting parallelism: sum over a tree



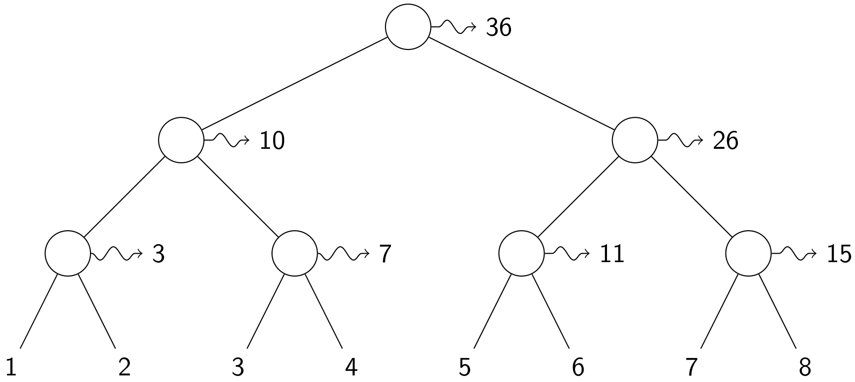
Exploiting parallelism: sum over a tree



Exploiting parallelism: sum over a tree

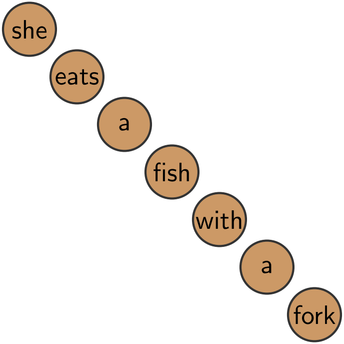


Exploiting parallelism: sum over a tree



- Picture a little computer at each node.
- The program "flows down" and the data "flows up".
- Computers of the future will have such a fractal structure.

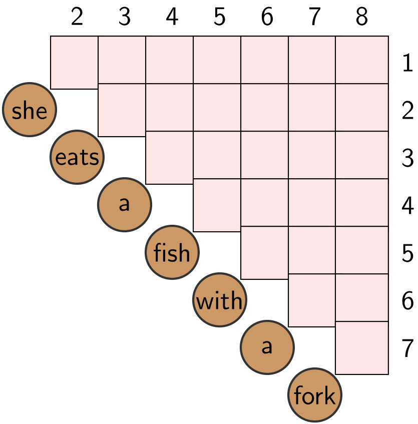
Chart parsing



$\mathcal{G}$ (CNF)

S	$\rightarrow$	NP VP
VP	$\rightarrow$	VP PP
VP	$\rightarrow$	VP NP
VP	$\rightarrow$	eats
PP	$\rightarrow$	P NP
NP	$\rightarrow$	Det N
NP	$\rightarrow$	she
P	$\rightarrow$	with
N	$\rightarrow$	fish
N	$\rightarrow$	fork
Det	$\rightarrow$	a

Chart parsing

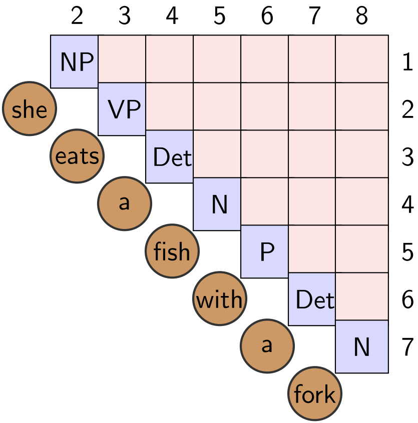


$\mathcal{G}$ (CNF)

S	$\rightarrow$	NP VP
VP	$\rightarrow$	VP PP
VP	$\rightarrow$	VP NP
VP	$\rightarrow$	eats
PP	$\rightarrow$	P NP
NP	$\rightarrow$	Det N
NP	$\rightarrow$	she
P	$\rightarrow$	with
N	$\rightarrow$	fish
N	$\rightarrow$	fork
Det	$\rightarrow$	a

R_{ij} = all non-terminals generating the input substring $w_{i..j}$

Chart parsing

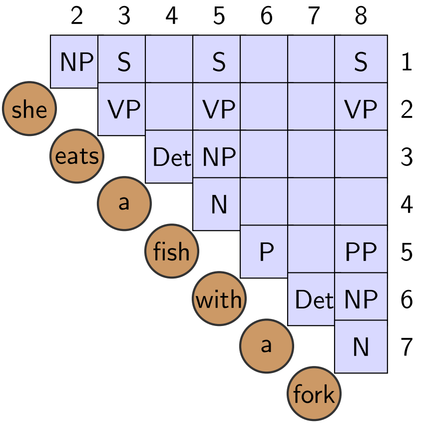


$\mathcal{G}$ (CNF)

S	$\rightarrow$	NP VP
VP	$\rightarrow$	VP PP
VP	$\rightarrow$	VP NP
VP	$\rightarrow$	eats
PP	$\rightarrow$	P NP
NP	$\rightarrow$	Det N
NP	$\rightarrow$	she
P	$\rightarrow$	with
N	$\rightarrow$	fish
N	$\rightarrow$	fork
Det	$\rightarrow$	a

$$R_{i,i+1} = \{A \mid A \rightarrow w_i \in \mathcal{G}\}$$

Chart parsing



$\mathcal{G}$ (CNF)

S	$\rightarrow$	NP VP
VP	$\rightarrow$	VP PP
VP	$\rightarrow$	VP NP
VP	$\rightarrow$	eats
PP	$\rightarrow$	P NP
NP	$\rightarrow$	Det N
NP	$\rightarrow$	she
P	$\rightarrow$	with
N	$\rightarrow$	fish
N	$\rightarrow$	fork
Det	$\rightarrow$	a

$$R_{i,i+1} = \{A \mid A \rightarrow w_i \in \mathcal{G}\}$$

$$R_{ij} = \{A \mid k \in [i+1..j-1], B \in R_{ik}, C \in R_{kj}, A \rightarrow BC \in \mathcal{G}\}$$

Parsing Specification (1)

Structure on sets of non-terminals:

$$x + y = x \cup y$$

$$x \cdot y = \{N \mid A \in x, B \in y, N \to AB \in \mathcal{G}\}$$

Parsing Specification (1)

Structure on sets of non-terminals:

$$x + y = x \cup y$$
$$x \cdot y = \{N \mid A \in x, B \in y, N \to AB \in \mathcal{G}\}$$

Structure on matrices:

$$(A + B)_{ij} = A_{ij} + B_{ij}$$
$$(A \cdot B)_{ij} = \sum_k A_{ik} \cdot B_{kj}$$

Parsing Specification (1)

Structure on sets of non-terminals:

$$x + y = x \cup y$$
$$x \cdot y = \{N \mid A \in x, B \in y, N \to AB \in \mathcal{G}\}$$

Structure on matrices:

$$(A + B)_{ij} = A_{ij} + B_{ij}$$
$$(A \cdot B)_{ij} = \sum_k A_{ik} \cdot B_{kj}$$

Is $\cdot$ associative?

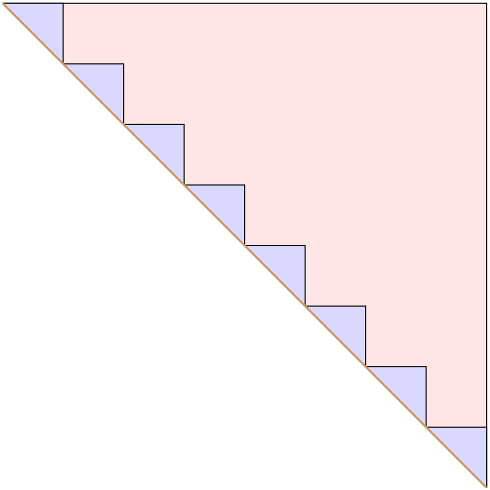
Parsing Specification (2)

Find smallest R, such that

$$R = I(w) + R \cdot R \tag{1}$$

$$(I(w))_{i,i+1} = \{A \mid A \rightarrow w_i \in \mathcal{G}\} \tag{2}$$

Chart parsing as Divide and Conquer



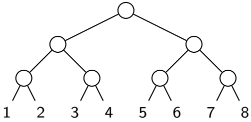
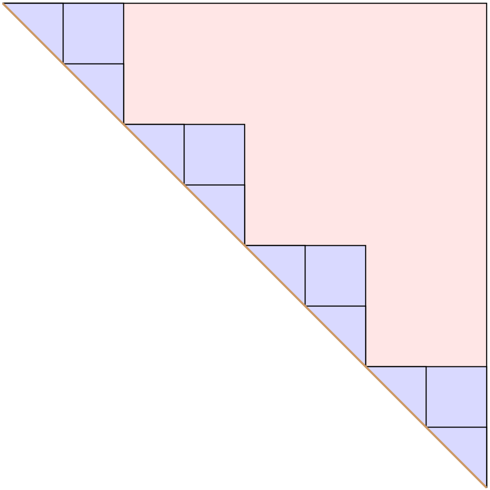


Chart parsing as Divide and Conquer



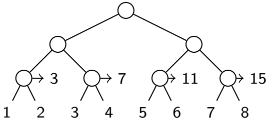
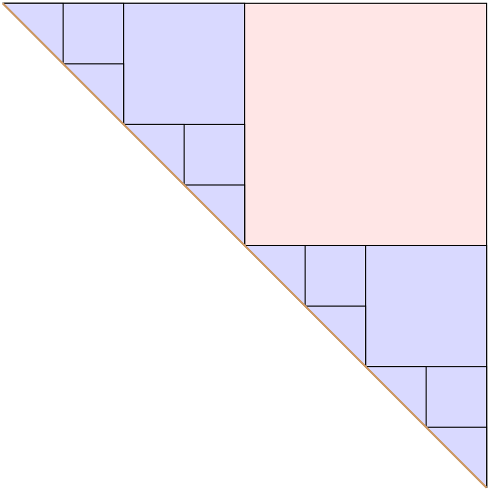


Chart parsing as Divide and Conquer



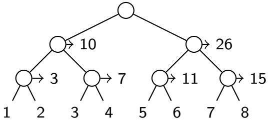
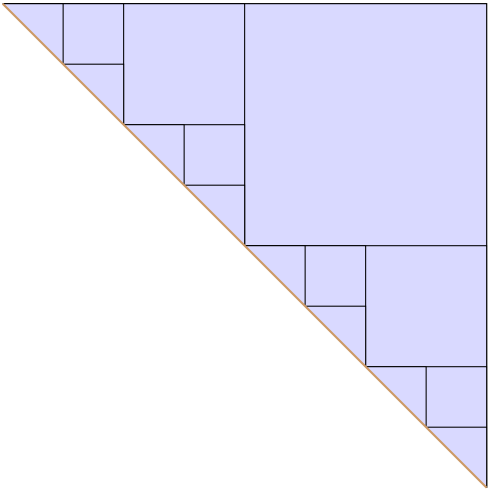
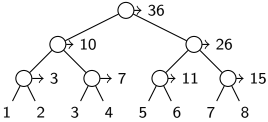


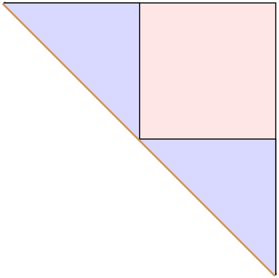
Chart parsing as Divide and Conquer





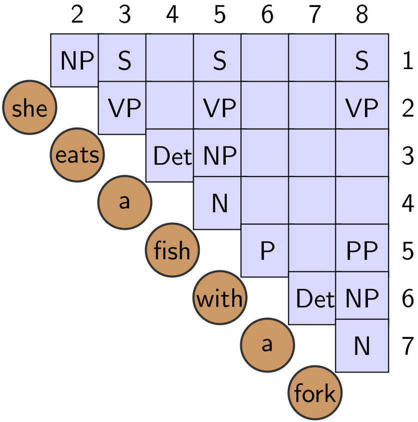
Efficiency

- space usage quadratic in the size of input string
- runtime cubic in the size of input string



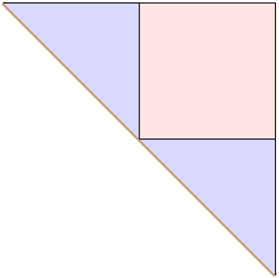
The combination operator takes cubic time

A sparse matrix



$$\#A \leq \left\lceil \alpha \sum_{(i,j)\in \mathrm{dom}(A)} \frac{1}{(j-i)^2} \right\rceil$$

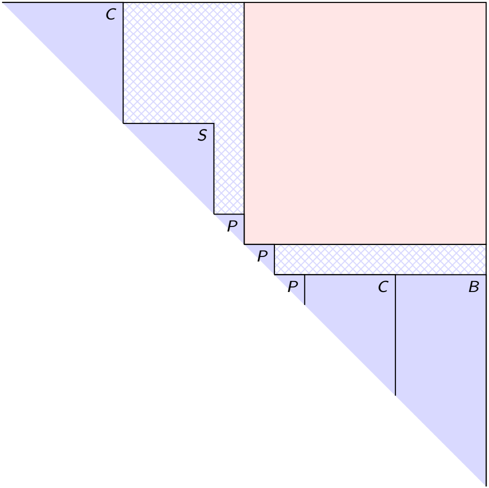
Cheap combination



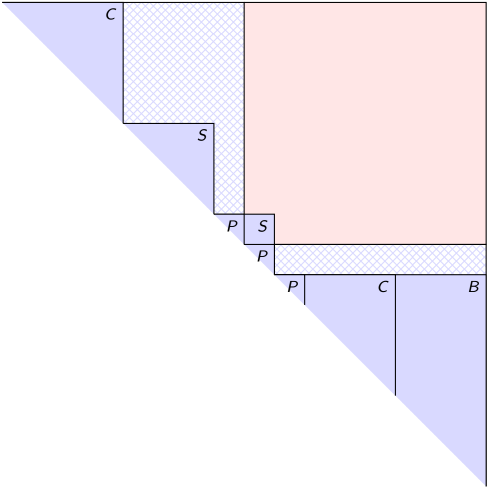
The square to fill is sparse

- To fill it should be quick
- Good space usage
- Good time-usage

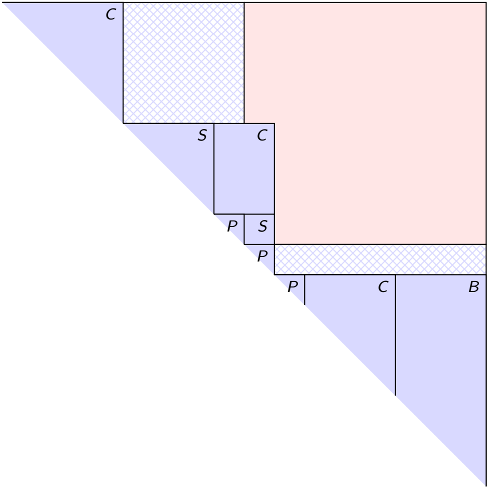
How much work?



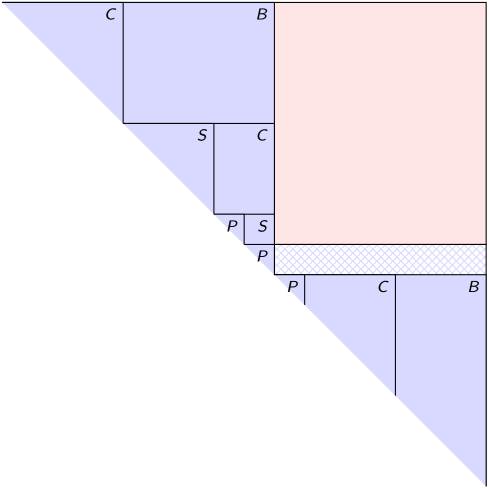
How much work?



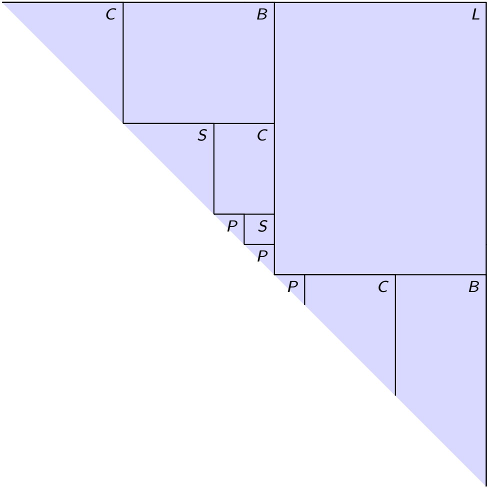
How much work?



How much work?



How much work?



Deriving Efficient Transitive Closure Algorithm

Problem: find R such that $R = R \cdot R + W$.

Deriving Efficient Transitive Closure Algorithm

Problem: find R such that $R = R \cdot R + W$.

$$W = \begin{bmatrix} A & X \\ 0 & B \end{bmatrix} \qquad R = \begin{bmatrix} A' & X' \\ 0 & B' \end{bmatrix}$$

Deriving Efficient Transitive Closure Algorithm

Problem: find R such that $R = R \cdot R + W$.

$$W = \begin{bmatrix} A & X \\ 0 & B \end{bmatrix} \qquad R = \begin{bmatrix} A' & X' \\ 0 & B' \end{bmatrix}$$

$$\begin{bmatrix} A' & X' \\ 0 & B' \end{bmatrix} = \begin{bmatrix} A' & X' \\ 0 & B' \end{bmatrix} \cdot \begin{bmatrix} A' & X' \\ 0 & B' \end{bmatrix} + \begin{bmatrix} A & X \\ 0 & B \end{bmatrix}$$

Deriving Efficient Transitive Closure Algorithm

Problem: find R such that $R = R \cdot R + W$.

$$W = \begin{bmatrix} A & X \\ 0 & B \end{bmatrix} \qquad R = \begin{bmatrix} A' & X' \\ 0 & B' \end{bmatrix}$$

$$\begin{bmatrix} A' & X' \\ 0 & B' \end{bmatrix} = \begin{bmatrix} A' & X' \\ 0 & B' \end{bmatrix} \cdot \begin{bmatrix} A' & X' \\ 0 & B' \end{bmatrix} + \begin{bmatrix} A & X \\ 0 & B \end{bmatrix}$$

$$\begin{aligned} A' &= A'A' + A \\ X' &= A'X' + X'B' + X \\ B' &= B'B' + B \end{aligned}$$

Deriving Efficient Chart Concatenation

Problem: find Y such that $Y = AY + YB + X = V(A, X, B)$.

Deriving Efficient Chart Concatenation

Problem: find Y such that $Y = AY + YB + X = V(A, X, B)$.

$$Y = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \quad X = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$$

$$A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix}$$

Deriving Efficient Chart Concatenation

Problem: find Y such that $Y = AY + YB + X = V(A, X, B)$.

$$Y = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \quad X = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$$

$$A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix}$$

$$\begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \cdot \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} + \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix} + \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$$

Deriving Efficient Chart Concatenation

Problem: find Y such that $Y = AY + YB + X = V(A, X, B)$.

$$\begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \cdot \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} + \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix} + \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$$

$$\begin{aligned} Y_{11} &= A_{11}Y_{11} + A_{12}Y_{21} + Y_{11}B_{11} + 0 + X_{11} \\ Y_{12} &= A_{11}Y_{12} + A_{12}Y_{22} + Y_{11}B_{12} + Y_{12}B_{22} + X_{12} \\ Y_{21} &= 0 + A_{22}Y_{21} + Y_{21}B_{11} + 0 + X_{21} \\ Y_{22} &= 0 + A_{22}Y_{22} + Y_{21}B_{12} + Y_{22}B_{22} + X_{22} \end{aligned}$$

Deriving Efficient Chart Concatenation

Problem: find Y such that $Y = AY + YB + X = V(A, X, B)$.

$$
\begin{aligned}
Y_{11} &= A_{11}Y_{11} + A_{12}Y_{21} + Y_{11}B_{11} + 0 + X_{11} \\
Y_{12} &= A_{11}Y_{12} + A_{12}Y_{22} + Y_{11}B_{12} + Y_{12}B_{22} + X_{12} \\
Y_{21} &= 0 + A_{22}Y_{21} + Y_{21}B_{11} + 0 + X_{21} \\
Y_{22} &= 0 + A_{22}Y_{22} + Y_{21}B_{12} + Y_{22}B_{22} + X_{22}
\end{aligned}
$$

$$
\begin{aligned}
Y_{11} &= A_{11}Y_{11} + X_{11} + A_{12}Y_{21} + Y_{11}B_{11} \\
Y_{12} &= A_{11}Y_{12} + X_{12} + A_{12}Y_{22} + Y_{11}B_{12} + Y_{12}B_{22} \\
Y_{21} &= A_{22}Y_{21} + X_{21} + 0 + Y_{21}B_{11} \\
Y_{22} &= A_{22}Y_{22} + X_{22} + Y_{21}B_{12} + Y_{22}B_{22}
\end{aligned}
$$

Deriving Efficient Chart Concatenation

Problem: find Y such that $Y = AY + YB + X = V(A, X, B)$.

$$
\begin{aligned}
Y_{11} &= A_{11}Y_{11} + X_{11} + A_{12}Y_{21} &&+ Y_{11}B_{11} \\
Y_{12} &= A_{11}Y_{12} + X_{12} + A_{12}Y_{22} + Y_{11}B_{12} &&+ Y_{12}B_{22} \\
Y_{21} &= A_{22}Y_{21} + X_{21} + 0 &&+ Y_{21}B_{11} \\
Y_{22} &= A_{22}Y_{22} + X_{22} + Y_{21}B_{12} &&+ Y_{22}B_{22}
\end{aligned}
$$

$$
\begin{aligned}
Y_{11} &= V(A_{11},\ X_{11} + A_{12}Y_{21}, && B_{11}) \\
Y_{12} &= V(A_{11},\ X_{12} + A_{12}Y_{22} + Y_{11}B_{12}, && B_{22}) \\
Y_{21} &= V(A_{22},\ X_{21}, && B_{11}) \\
Y_{22} &= V(A_{22},\ X_{22} + Y_{21}B_{12}, && B_{22})
\end{aligned}
$$

Deriving Efficient Chart Concatenation

Problem: find Y such that $Y = AY + YB + X = V(A, X, B)$.

$$Y = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \quad X = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$$

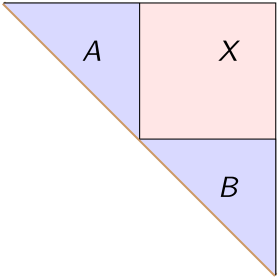
$$A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix}$$

$$\begin{aligned}
Y_{11} &= V(A_{11},\ X_{11} + A_{12}Y_{21}, \ B_{11}) \\
Y_{12} &= V(A_{11},\ X_{12} + A_{12}Y_{22} + Y_{11}B_{12},\ B_{22}) \\
Y_{21} &= V(A_{22},\ X_{21}, \ B_{11}) \\
Y_{22} &= V(A_{22},\ X_{22} + Y_{21}B_{12}, \ B_{22})
\end{aligned}$$

No circular dependencies! Done!

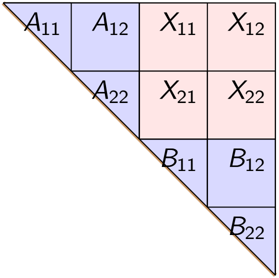
Valiant's algorithm for transitive closure

$$Y = V(A, X, B)$$



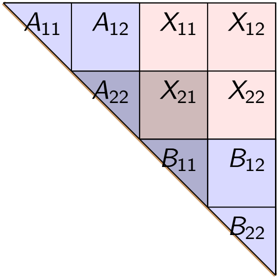
Valiant's algorithm for transitive closure

$$Y = V(A, X, B)$$



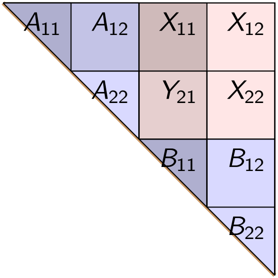
Valiant's algorithm for transitive closure

$$Y = V(A, X, B)$$



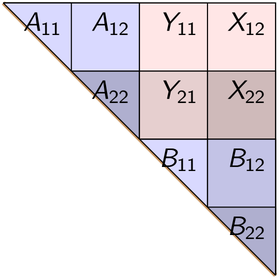
Valiant's algorithm for transitive closure

$$Y = V(A, X, B)$$



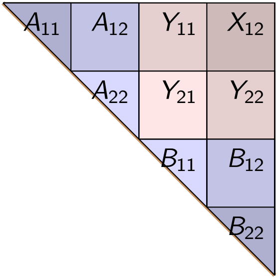
Valiant's algorithm for transitive closure

$$Y = V(A, X, B)$$



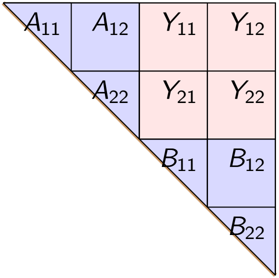
Valiant's algorithm for transitive closure

$$Y = V(A, X, B)$$



Valiant's algorithm for transitive closure

$$Y = V(A, X, B)$$



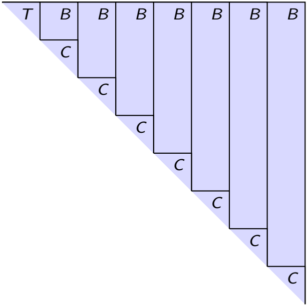
Haskell Implementation: Sparse Matrix Structure

$$
\begin{array}{l}
\textbf{import } \mathit{Prelude}\ (\mathit{Eq}\ (..)) \\
\textbf{class } \mathit{RingLike}\ a \textbf{ where} \\
\quad \mathit{zero} :: a \\
\quad (+) :: a \to a \to a \\
\quad (\cdot) :: a \to a \to a \\
\textbf{data } M\ a = Q\ (M\ a)\ (M\ a)\ (M\ a)\ (M\ a) \mid Z \mid \mathit{One}\ a \\
q\ Z\ Z\ Z\ Z = Z \\
q\ a\ \ b\ \ c\ \ d = Q\ a\ b\ c\ d \\
\mathit{one}\ x = \textbf{if } x \equiv \mathit{zero} \textbf{ then } Z \textbf{ else } \mathit{One}\ x
\end{array}
$$

Haskell Implementation: algorithm

$$
\begin{array}{llll}
\multicolumn{4}{l}{\textbf{instance } (Eq\ a, RingLike\ a) \Rightarrow RingLike\ (M\ a) \textbf{ where} \quad \text{-- ...}} \\
\multicolumn{4}{l}{v :: (Eq\ a, RingLike\ a) \Rightarrow M\ a \to M\ a \to M\ a \to M\ a} \\
v\ a & Z & b = Z & \\
v\ Z & (One\ x) & Z = One\ x & \\
\multicolumn{4}{l}{v\ (Q\ a_{11}\ a_{12}\ Z\ a_{22})\ (Q\ x_{11}\ x_{12}\ x_{21}\ x_{22})\ (Q\ b_{11}\ b_{12}\ Z\ b_{22})} \\
\multicolumn{4}{l}{\quad = q\ y_{11}\ y_{12}\ y_{21}\ y_{22}} \\
\multicolumn{4}{l}{\qquad \textbf{where } y_{21} = v\ a_{22}\ \ x_{21} \qquad\qquad\qquad\qquad\qquad\qquad\qquad b_{11}} \\
\multicolumn{4}{l}{\qquad\qquad\quad y_{11} = v\ a_{11}\ (x_{11} + a_{12} \cdot y_{21} \qquad\qquad\qquad)\ b_{11}} \\
\multicolumn{4}{l}{\qquad\qquad\quad y_{22} = v\ a_{22}\ (x_{22} + \qquad\qquad\qquad y_{21} \cdot b_{12})\ b_{22}} \\
\multicolumn{4}{l}{\qquad\qquad\quad y_{12} = v\ a_{11}\ (x_{12} + a_{12} \cdot y_{22} + y_{11} \cdot b_{12})\ b_{22}}
\end{array}
$$

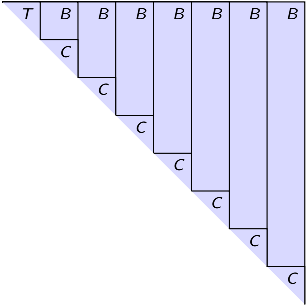
Recursion in the grammar



$B \rightarrow TitlePage$

$B \rightarrow BC$

Recursion in the grammar

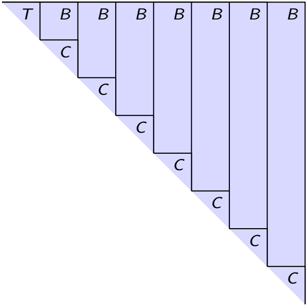


$B \to TitlePage$

$B \to BC$

Bad!

Recursion in the grammar



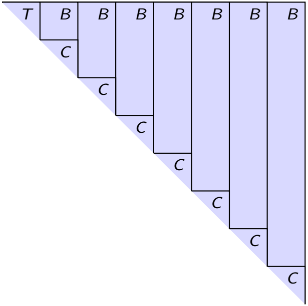
$B \rightarrow TitlePage$

$B \rightarrow BC$

Bad!

- The combination has a lot of work to do (at least linear)

Recursion in the grammar



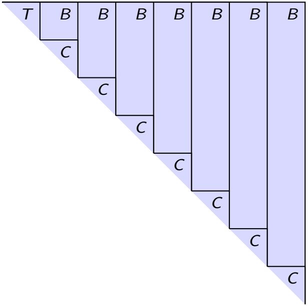
$B \rightarrow TitlePage$

$B \rightarrow BC$

Bad!

- The combination has a lot of work to do (at least linear)
- AST is a list

Recursion in the grammar



$B \rightarrow \mathit{TitlePage}$

$B \rightarrow BC$

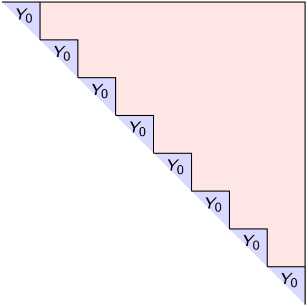
Bad!

- The combination has a lot of work to do (at least linear)
- AST is a list

Solution: $B' \rightarrow \mathit{TitlePage}\ B$

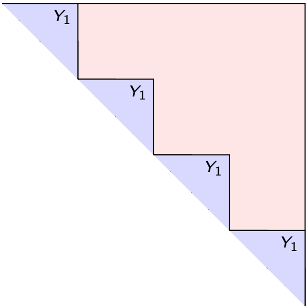
$B \rightarrow C*$

Binary encoding of lists: idea



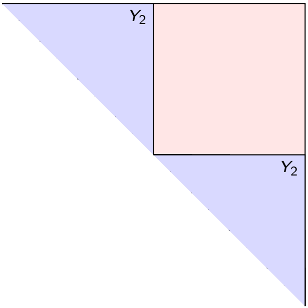
$L \to Y*$

Binary encoding of lists: idea



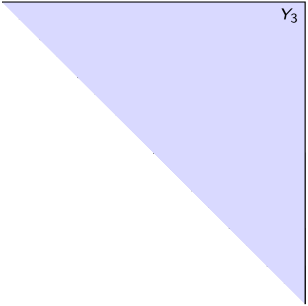
$L \to Y*$

Binary encoding of lists: idea



$L \to Y*$

Binary encoding of lists: idea



$L \to Y*$

Summary

- Valiant (75) does parsing using matrix multiply; yields the most efficient known CF recognition algorithm: $O(n^{2.3727})$. [1]
- The *very same* algorithm yields parsing on "good" inputs in $O(n)$. The conquer step costs $O(log^2 n)$ (instead of $O(n^{2.3727})$).
- Simple, effective, flexible algorithm.
- Implemented in BNFC: push-button technology.
- It *is* fast: Incremental parsing of a 8000-line C program in less than 1 millisecond.

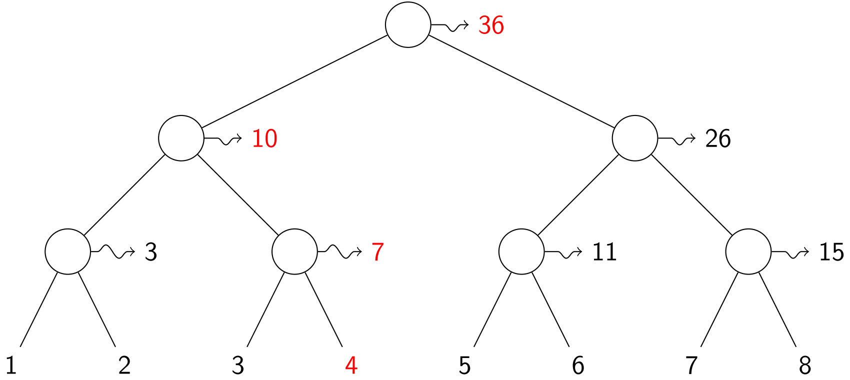
Full details: `http://cse.chalmers.se/~bernardy/PP.pdf`

[1] Complexity of the Coppersmith-Winograd algorithm

Perspective

- Works with any non-associative ring-like structure. The same algorithm supports probabilistic parsing; context sensitive; etc. (To be efficient we need a cutoff heuristic though.)
- Maybe suitable for deep learning?
- "High level" cells activated (much more) seldom

BONUS: Incremental computation (1)



BONUS: Incremental computation (2)